Class 6: R Functions

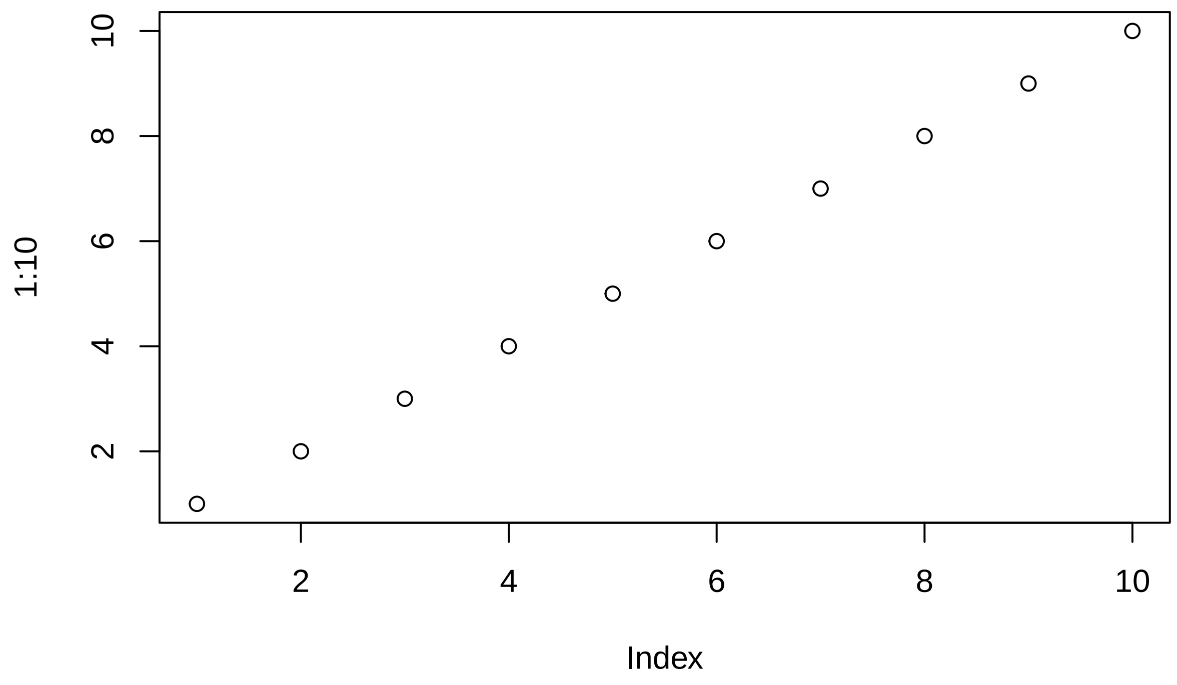
Anu Chaparala (PID: A15484707)

10/14/2021

A play with Rmarkdown

This is some plain text. I can make things **bold**. I can also make things *italic*.

```
#insert R coding area/chunk using small green c+ button in top bar
plot(1:10)
```



R functions

In today's class we are going to write a function together that grades some students' work.

Questions for today:

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)
```

Let's start with student1 and find their average score.

```
mean(student1)
```

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

But we want to drop the lowest score... We could try the **min()** function.

```
min(student1)
```

```
## [1] 90
```

```
#which.min gives location of minimum value within the vector
```

The **which.min** function looks useful:

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

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

```
#so, student 1's lowest score is in element 8 of the student1 vector
```

Now we will use this to removed student 1's lowest score from the original vector.

```
#this would be the lowest score
student1[which.min(student1)]
```

```
## [1] 90
```

To drop this value, I can use minus.

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

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

Let's now use mean() to get the average minus the lowest score.

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

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

```
#success!
```

Let's look at student2.

```
student2
```

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

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

```
## [1] NA
```

```
#our mean function from earlier does not work with this function b/c of the "NA" string. Function gives
```

We need to remove the NA elements of the vector

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

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

```
mean(student2[-which.min(student2)], na.rm=TRUE)
```

```
## [1] 92.83333
```

This is not what we want. It dropped the 80 (i.e. the lowest number and not the NA i.e. missing hw).

Let's look at student3.

```
student3
```

```
## [1] 90 NA NA NA NA NA NA NA
```

```
mean(student3[-which.min(student3)], na.rm=TRUE)
```

```
## [1] NaN
```

One new idea/approach is we could replace the NA (missing hwks) with zero.

Let's try this with student2.

```
student2
```

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

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

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

The **is.na** function returns a logical vector where TRUE elements represent where the NA values are. We can also put which(is.na) to get exact positions within the vector.

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

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

So, now let's make the NA values into zeros.

```r
student.prime <- student2
student.prime
```

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

```r
student.prime[which(is.na(student.prime))] = 0
student.prime
```

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

```r
#or we can replace the NA score using the replace() function
replace(student2, which(is.na(student2)), 0)
```

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

Now we need to put this all together to get the average score, dropping the lowest where we map NA values to 0.

```r
student.prime <- student2
student.prime
```

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

```r
student.prime[which(is.na(student.prime))] = 0
student.prime
```

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

```r
mean(student.prime[-which.min(student.prime)])
```

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

```r
mean(c(100, 90, 90, 90, 90, 97, 80))
```

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

```r
#it worked!
```

Let's check and make sure this works with student3

```r
student.prime <- student3
student.prime
```

```
## [1] 90 NA NA NA NA NA NA NA
```

```r
student.prime[which(is.na(student.prime))] = 0
student.prime
```

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

```r
mean(student.prime[-which.min(student.prime)])
```

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

```r
#success!
```

We got out working snippet! Let's simplify.

```r
x <- student3
#Map NA value to zero.
x[which(is.na(x))] = 0
#Find the mean without the lowest score.
mean(x[-which.min(x)])
```

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

Now we can use this as the body of my function.

```r
grade <- function(x) {
  #Make sure our scores are all numbers
  x <-as.numeric(x)

#Map NA value to zero.
x[which(is.na(x))] = 0
#Find the mean without the lowest score.
mean(x[-which.min(x)])

}
```

```r
#successful function!
grade(student1)
```

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

```r
grade(student2)
```

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

```r
grade(student3)
```

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

Now we will apply ou function to an entire gradebook.

```r
#Read the full gradebook CSV file.
scores <- read.csv("https://tinyurl.com/gradeinput", row.names=1)
scores
```

```
##            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
```

```r
scores[1,]
```

```
##           hw1 hw2 hw3 hw4 hw5
## student-1 100  73 100  88  79
```

Check if function works on individual rows from CSV.

```
grade(as.numeric(scores[1,]))
```

```
## [1] 91.75
```

```
grade(scores[10,])
```

```
## [1] 79
```

Examples of using function **is.numeric**

```
is.numeric(student1)
```

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

```
is.numeric(scores[10,])
```

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

```
as.numeric(c(1,2,NA,4,5))
```

```
## [1]  1  2 NA  4  5
```

Now grade all students by using the **apply()** function.

```
apply(scores,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
```

```
ans2 <- apply(scores,1, grade)
```

> Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

```
which.max(ans2)
```

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

Student 18 has the overall top score in the gradebook, at 94.50.

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

We can use the apply() function over the columns by seeting the margin=2 argument.

```r
ans3 <- apply(scores, 2, mean, na.rm=TRUE)
ans3
```

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

```r
which.min(ans3)
```

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

Based on this application, hw3 appears to be the toughest assignment for the students, with average score of 80.80.

> 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]