Class 6: R Functions

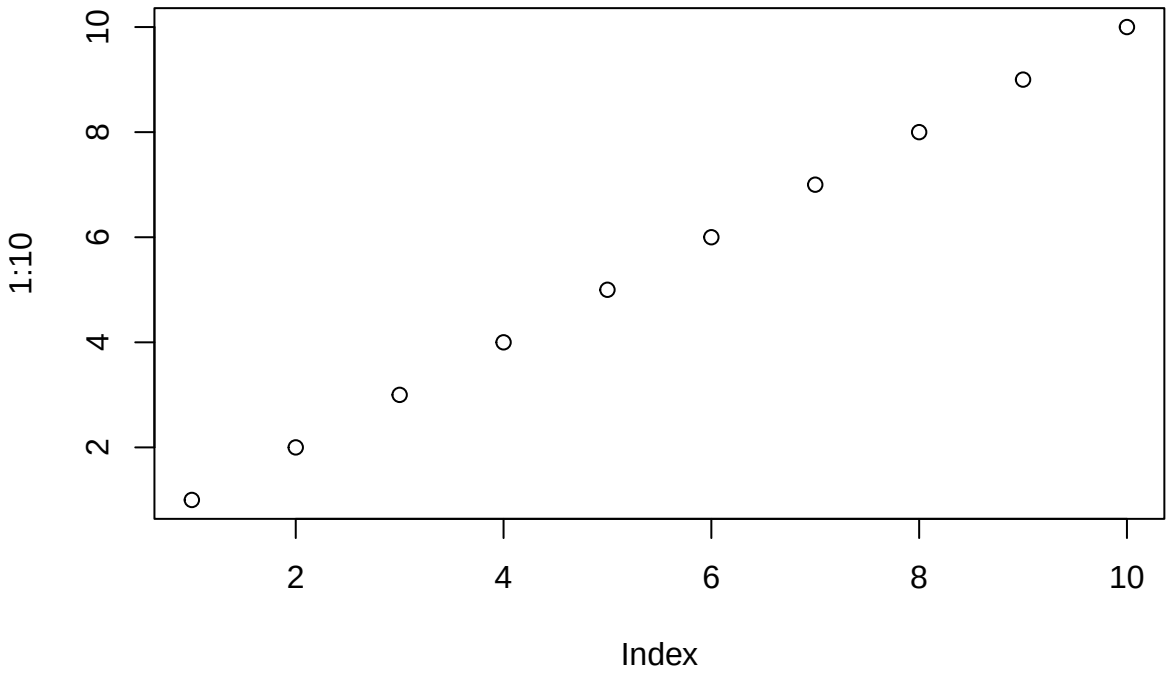
Tasnia Sharia (PID A15931128)

10/14/2021

A play with Rmarkdown

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

```
# This is a code chunk
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's:

> **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
```

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

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

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

This gives the position of the lowest score.

```
# This would be the lowest score for student1
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
```

Now we can find the mean() with the lowest score dropped

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

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

Now let's try student2 grades

```
student2
```

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

This did not work cause of the NA

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

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

We need to remove the NA elements of the student2 vector

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

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

It dropped the 80(the lowest score) but not the NA(missing homework). Need to try something else

Let's try 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 homework) with zero

Let's go back to student2

```
student2
```

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

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

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

The **is.na()** functions returns a logical vector where TRUE elements represent where the NA values are positioned.

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

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

Now let's make the NA values into zeros

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

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

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

```
## [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 score where they map the NA values to zero.

```
student.prime <- student2
student.prime[ which(is.na(student.prime)) ] = 0
mean(student.prime[ -which.min(student.prime) ])
```

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

Let's check if it works for student3

```
student.prime <- student3
student.prime[ which(is.na(student.prime)) ] = 0
mean(student.prime[ -which.min(student.prime) ])
```

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

We got our working snippet! Let's simplify.

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

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

Now we can use this as the body of my functions and create grade()

```
grade <- function(x) {
  # Make sure our scores are all numbers
  x <- as.numeric(x)
  # Map NA values to zero
  x[ which(is.na(x)) ] = 0
  # Find the mean without the lowest value
  mean(x[ -which.min(x) ])
}
```

```
grade(student2)
```

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

Now read the full grade book 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
```

Making the NA inputs into numeric values. Had to change the grade() by adding the as.numeric()

```
grade(as.numeric(scores[10,]))
```

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

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

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

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

```
ans <- apply(scores,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]

```
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]

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

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

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

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

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

Q5. Make sure you save your Rmarkdown document and can click the "Knit" button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope. [1pt]

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]

```
y <- scores
# maps NA values to zero
y[is.na(y)] = 0
y
```

```
##            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
```

```
cor(ans, scores$hw1)
```

```
## [1] 0.4250204
```

Applying correlation to each hw

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

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

Finding the highest correlation

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

```
## hw5
##   5
```