Class06: R Functions

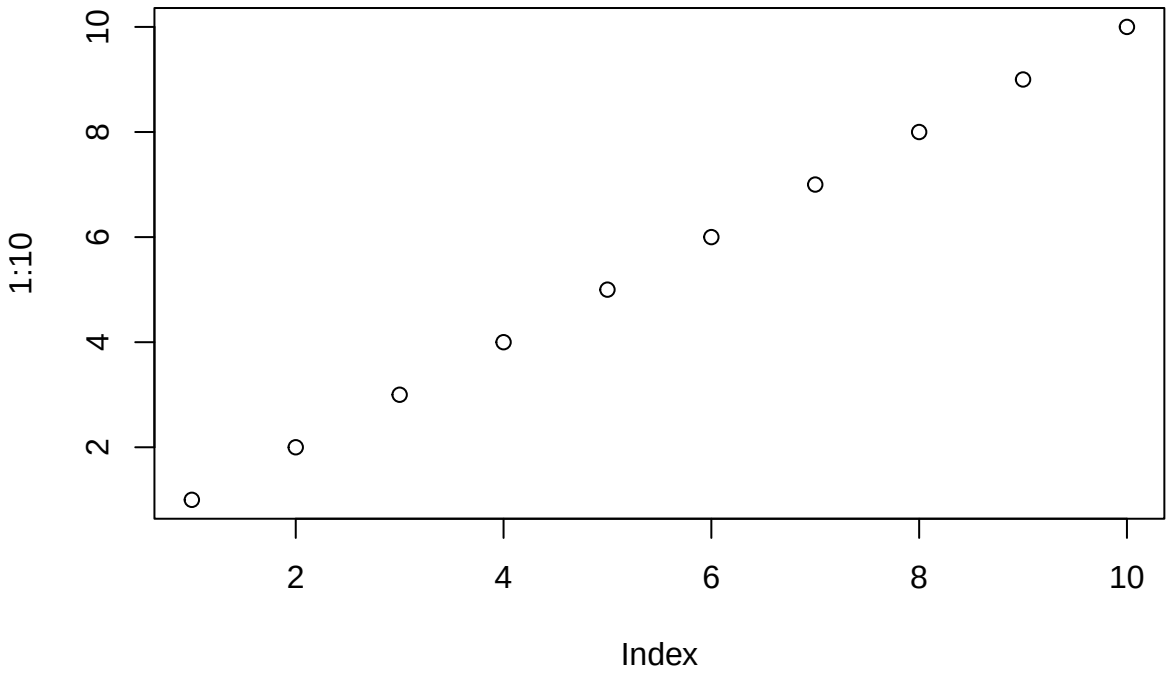
Monica Lin (PID: A15524235)

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 an R 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:

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]

ctrl+alt+i is a keyboard shortcut to call up R code chunk! ctrl+return runs the command.

```
# 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 numeric score

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

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

To drop this value, use minus to print out all the other scores

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

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

Now use **mean()** to get the average minus the lowest score

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

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

Let's try this with student2

```
student2
```

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

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

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

student2 would get NA as their average grade, just for one missing homework :-(We need to remove the NA elements of the vector

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

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

This is still not what we want. It dropped the 80 (i.e. the lowest number) instead of the NA (i.e. the missing homework).

Let's look at student3

```
student3
```

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

na.rm=TRUE would remove all of the NAs and returns NaN (i.e. Not a Number)

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

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

One approach to solve this is to replace any NAs with zeroes.

Let's try with 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()** function returns a logical vector where TRUE elements represent positions of NA values.

which(is.na(x)) gives the position of NA

```
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 where we map NA values to zero.

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

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

```
student.prime
```

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

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

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

Looks good! Check 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 our function.

```
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(student3)
```

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

The function works!

Now 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
```

```
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
```

Use for one student

```
grade(scores[1,])
```

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

Change scores to numeric values by altering previous code. **x <- as.numeric(x)** to avoid having to change each one individually via **grade(as.numeric(scores[2,]))** Row names must also be set as numeric values, otherwise "student-X" will be read as hw data. **row.names=1**

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
```

```
ans <- 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(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. To use the apply() function over the rows, set margin=1.

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

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

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

HW3 was toughest.

> **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]

```
mean(ans)
```

```
## [1] 87.425
```

```
difficult - mean(ans)
```

```
##       hw1       hw2       hw3       hw4       hw5
##  1.575000 -6.536111 -6.625000  2.206579 -4.003947
```

HW1 was most predictive of overall score because HW1 returns a value closest to 0, meaning that homework average is least different from the class average.

Alternate method to answer **Q4**.

```
mask <- scores
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
```

```
cor(ans, mask$hw3)
```

```
## [1] 0.3042561
```

Do for all with apply

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

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

Using the correlation method, HW5 gives the highest value, so HW5 correlates most strongly and is most predictive of overall score.