Class 6: R Functions

Wai Lam Adele Hong A15999023

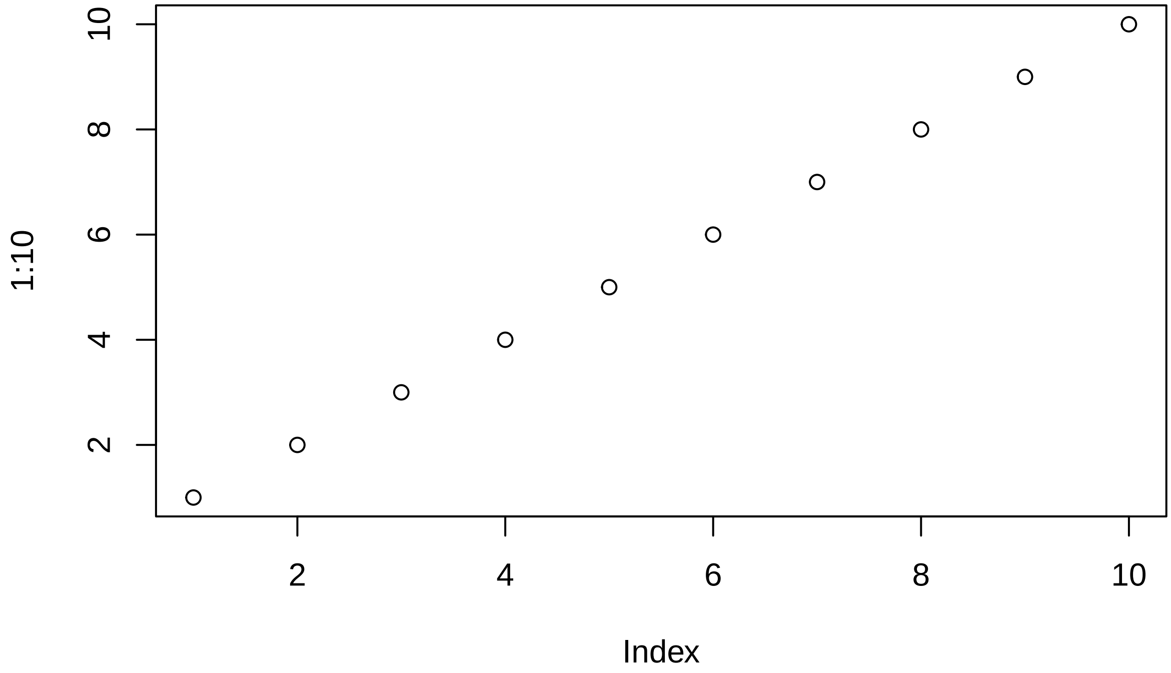
10/14/2021

A play with Rmarkdown

(1 hashtag is level 1 heading, 2 hashtags is level 2, etc.)

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

```r
# 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 student work.

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”

Data set

```
# 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
```

Let’s try the **which.min()** function.

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

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

```
# This gives the position of the lowest score.
```

We want to omit the lowest score.

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

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

Now let’s use mean() to get the average minus the lowest score.

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

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

Let’s do student2 now.

```r
student2
```

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

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

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

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

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

The code for student1 didn't work for student2 bc of "NA" in data. By using na.rm=TRUE, the NA in data set is removed. We recognize that there's a problem in which.min(student2) bc this gives us the position of the smallest **numeric** value, so it disregards the NA.

One idea is we could replace the NA with 0.

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

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

```r
# This gives us a logical vector of whether each value is "NA".

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

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

```r
# This helps us find **which** position the TRUE values are.

# Now let's make NA=0
student2.prime <- student2
student2.prime[which(is.na(student2.prime))]=0
student2.prime
```

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

```r
# We want to use a new variable "student.prime" bc if we just use "student.prime[which(is.na(student2))]
```

Now let's omit the lowest score from student2.

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

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

It works!!!

Moving on to student3.

```
student3.prime <- student3
student3.prime[which(is.na(student3.prime))]=0
mean(student3.prime[-which.min(student3.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) {
# The table provided is not numeric. Make sure our scores are all numbers; without this, we will get an
  x <- as.numeric(x)
  # NOTE: NA's will still be NA's, but numbers are just turned numeric. :-)).

# 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 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
```

```
# The "rownames=1" tells the data frame that the first column (col=1) is the row name for each row of H
```

Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? Now grade all students by using the **apply()** function.

```
z <- apply(scores, 1, grade)
# The "1" in the middle refers to the rows. A number of "2" refers to the columns.
which.max(z)
```

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

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

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

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

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)?

Plan: Finding the difference between the average student scores and use **cor()** function.

```
#scores
#z
cor(z, scores$hw1)
```

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

Now what about more homeworks?

```
# Replace all NA values with zero
scores[is.na(scores)] = 0
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   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(z, scores$hw2)
```

```
## [1] 0.176778
```

We can use the apply() function to do this for all columns of scores (i.e. every homework)

```
apply(scores,2, cor, z)
```

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

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