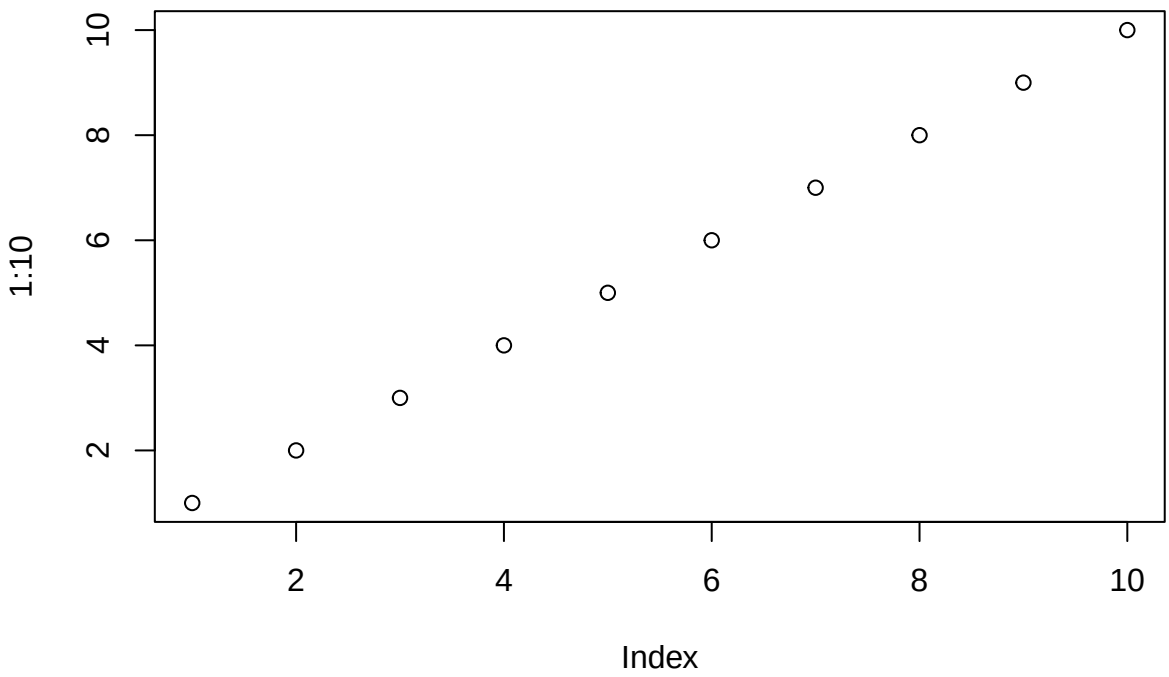
class06: R Functions

Jocelyn Olvera

10/15/2021

```
plot(1:10)
```



R Markdown

R Functions Lab Here we will write a function to grade some student homework.

We will start with a more simple input example - a vector of student homework scores.

```
# 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)
```

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]

To reveal the scores of student 1, just call for 'student1'.

```
student1
```

```
## [1] 100 100 100 100 100 100 100  90
```

The regular average will be returned by the 'mean()' function.

```
mean(student1)
```

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

Finding the lowest score using 'min()' function.

```
min(student1)
```

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

To find the position of the smallest value (i.e min) value in our vector we can use 'which.min()' function.

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

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

This calls for a score at a given position [ ].

```
student1[8]
```

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

Another function to find the min value.

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

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

To get everything but the min value use the minus sign (-).

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

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

Then take the mean for this vector.

```
## This is my first solution/snipet
mean(student1[-which.min(student1)])
```

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

Using 'na.rm=TRUE' argument will drop NA from the mean calculation. This makes it unfair when compared to students that completed all homework assignments.

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

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

Instead, map/change the NA values to zero for a fair comparison. Use the 'is.na()' function to call for NA.

```
student2
```

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

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

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

Creating a shortcut for 'student2' and changing NA to 0.

```
x <- student2
x
```

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

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

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

```
mean(x)
```

```
## [1] 79.625
```

Combining our working snippets to find the average score for student3.

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

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

Q1 Work: Grade Function Taking the working snipped and making it into a function. Go to **"Code"** tab and select **"Extract Function".**

```
grade <- function(x) {
  x[is.na(x)] <- 0
  mean(x[-which.min(x)])
}
```

Using the Function for student grades.

Student 1

```
grade(student1)
```

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

Student 2

```
grade(student2)
```

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

Student 3

```
grade(student3)
```

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

To better annotate your code, go to the **"Code"** tab and select **"Insert Roxygen Skeleton"**

```
#' Calculate average scores for a vector of homework scores.
#' Dropping the lowest single score. Missing values
#' will be treated as a zero score.
#' @param x Numeric vector of homework scores.
#'
#' @return Average score.
#' @export
#'
#' @examples
#' Students <- c(100, NA, 90, 80)
#' grade(student)
grade <- function(x) {
  # Map NA missing homework values to zero.
  # Missing homeworks score zero.
  x[is.na(x)] <- 0
  # We exclude lowest scored homework.
  mean(x[-which.min(x)])
}
```

```
# First load the CSV file.
url <- "https://tinyurl.com/gradeinput"
# Make the student names as first column.
gradebook <- read.csv(url, row.names=1)
gradebook
```

Now we can take the gradebook and grade the whole class of multiple students.

```
##            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
```

Use **'apply()'** to grade all the students with the **'grade()'** function.

```
apply(gradebook, 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
```

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

```
# First view the results.
results <- apply(gradebook, 1, grade)

# Sort the results in increasing order.
sort(results)
```

```
## student-15 student-10  student-2 student-19 student-20  student-3  student-4
##      78.75      79.00      82.50      82.75      82.75      84.25      84.25
## student-11  student-9 student-14 student-17  student-5  student-6 student-16
##      86.00      87.75      87.75      88.00      88.25      89.00      89.50
##  student-1 student-12 student-13  student-8  student-7 student-18
##      91.75      91.75      92.25      93.75      94.00      94.50
```

```
which.max(results)
```

Q2 Work: Top Scoring Student

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

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

```
# First, we are trying the mean stat first for each homework assignments.
hw.ave <- apply(gradebook, 2, mean, na.rm=TRUE)
# Call for the homework assignment with the lowest mean.
which.min(hw.ave)
```

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

Q3 Work: Toughest Homework

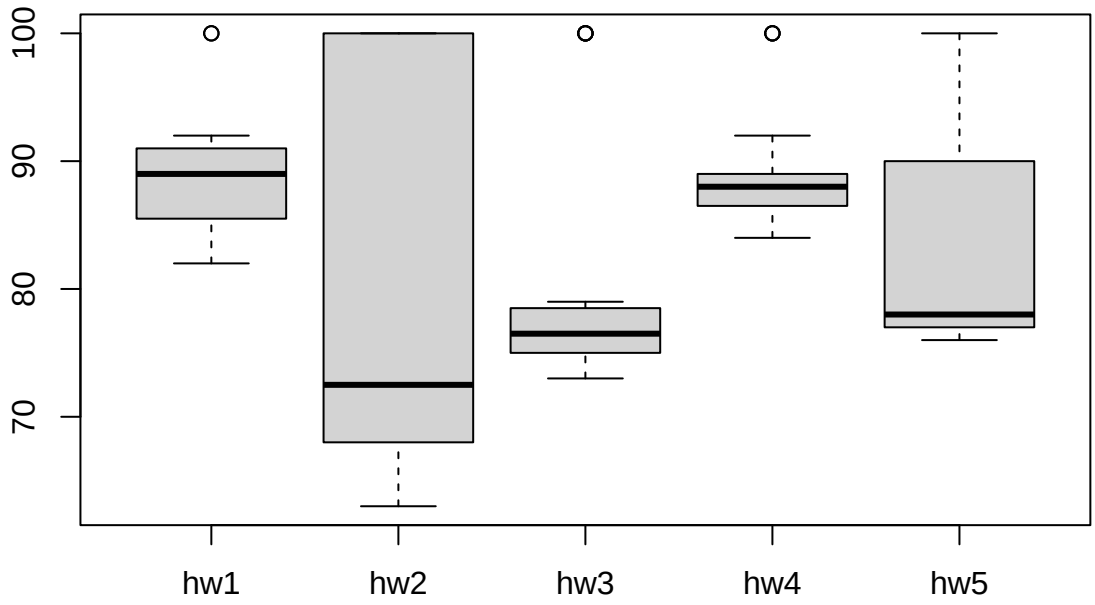
- Use the median stat instead of mean in order to get a better representation of which homework was hardest on students. This gives a better view of the distribution of scores. *

```
# Find the median for each homework assignment.
hw.med <- apply(gradebook, 2, median, na.rm=TRUE)
# Call for lowest median.
which.min(hw.med)
```

```
## hw2
##   2
```

Making a box plot of the student gradebook data.

```
boxplot(gradebook)
```



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]

Use the **cor()** function to find he most predictive score.

```
# First assign all NA values to zero.
gradebook[ is.na(gradebook)] <- 0
# Use the 'cor()' function to find the correlation value for a specific homework assignment '$hw'.
cor(results, gradebook$hw1)
```

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

```
# The 'apply()' function selects for all the columns '2' and gives the correlation 'cor'.
apply(gradebook, 2, cor, x=results)
```

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

Students performed worse on homework 2.