lab 6 R functions

Nadia Haghani

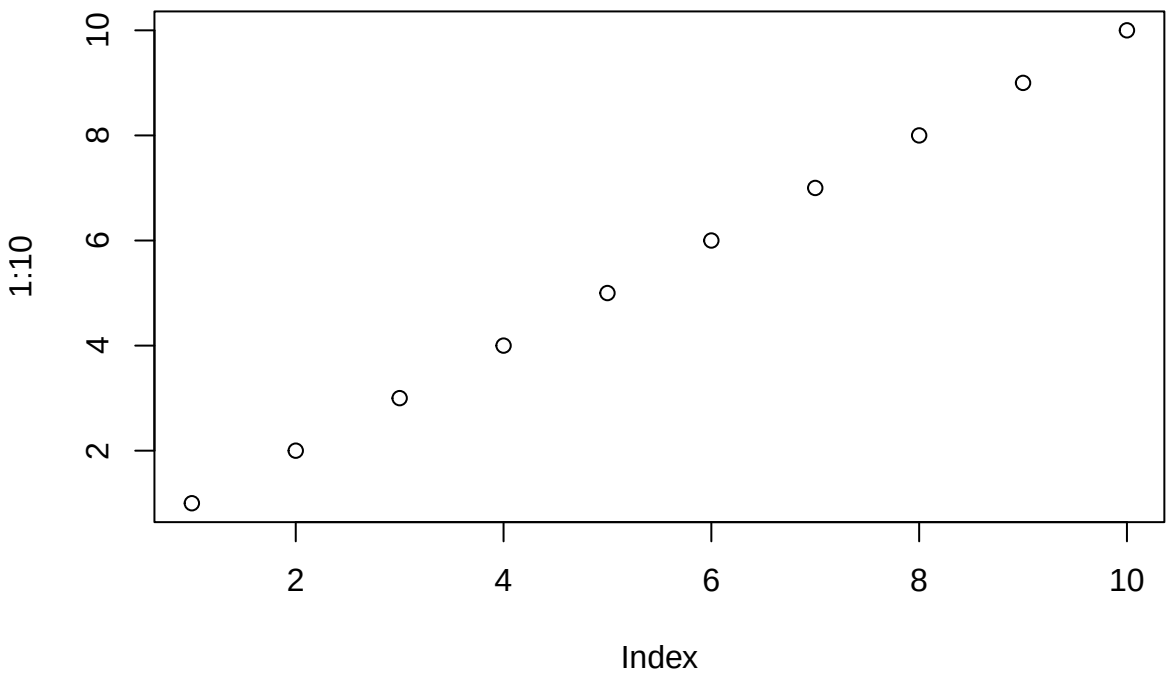
10/18/2021

This week we are introducing **R functions** and how to write our own R functions.

Questions to answer

> Q1: 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]

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



```
# 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)
```

Follow the guidelines from class - Write a working snipet of code that solves a simple problem

```
#Straight forward mean
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)

mean(student1)
```

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

But we need to drop the lowest score, or the minimum value

```
# Which element of the vector is the lowest?
which.min(student1)
```

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

What I want is to now exclude the lowest score from my mean calculation

```
# This will return everything but the 8th element in the vector
student1[-8]
```

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

We can use the answer from which.min to return all other elements of the vector

```
# This is our first working snipet
mean(student1[-(which.min(student1))])
```

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

What about the other students? Will this work for them?

We could try using na.rm is TRUE, but it is unfair

```
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
mean(student2, na.rm=TRUE)
```

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

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
mean(student3, na.rm=TRUE)
```

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

Another approach is to mask(replace) all NAs with zero

```
x <- student2
which(is.na(x))
```

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

Now we identified NA element positions we want to remove. We want to replace NA with zero

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

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

```
mean(x)
```

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

Drop the lowest score

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

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

Now for student 3.

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
x <- student3
x[is.na(x)] <- 0
mean(x[-(which.min(x))])
```

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

Now we make our function

Take snippet and turn into a function. Each function has 3 parts

- A name, in our case 'grade()'
- input arguments, a vector of student scores
- The body, our working snippet

Using RStudio, I will select 'Code > Extract Function'

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

```
grade(student1)
```

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

```
grade(student2)
```

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

```
grade(student3)
```

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

This looks great! We need to add comments to explain this to our future selves and others who want to use this function.

```
#' Calculate average score for a vector of student homework scores
#' dropping the lowest score.
#' Missing values are treated as zero.
#'
#' @param x A numeric vector of homework scores
#'
#' @return Average score
#' @export
#'
#' @examples
#' student <- c(100, NA, 90, 97)
#' grade(student)
#'

grade <- function(x) {
  # mask NA with zero
  # Treat missing values as zero
  x[is.na(x)] <- 0
  # Exclude lowest score from mean
  mean(x[-(which.min(x))])
}
```

Now finally we can use our funtino on our "real" whole class data from this CSV format file: "https://tinyurl.com/gradeinput"

```
url <- "https://tinyurl.com/gradeinput"
gradebook <- read.csv(url, row.names=1)
```

```
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]

To answer this, we run the apply() function and save the results.

```
results <- apply(gradebook, 1, grade)
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)
```

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

So, the highest top scoring student is student 18.

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

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

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

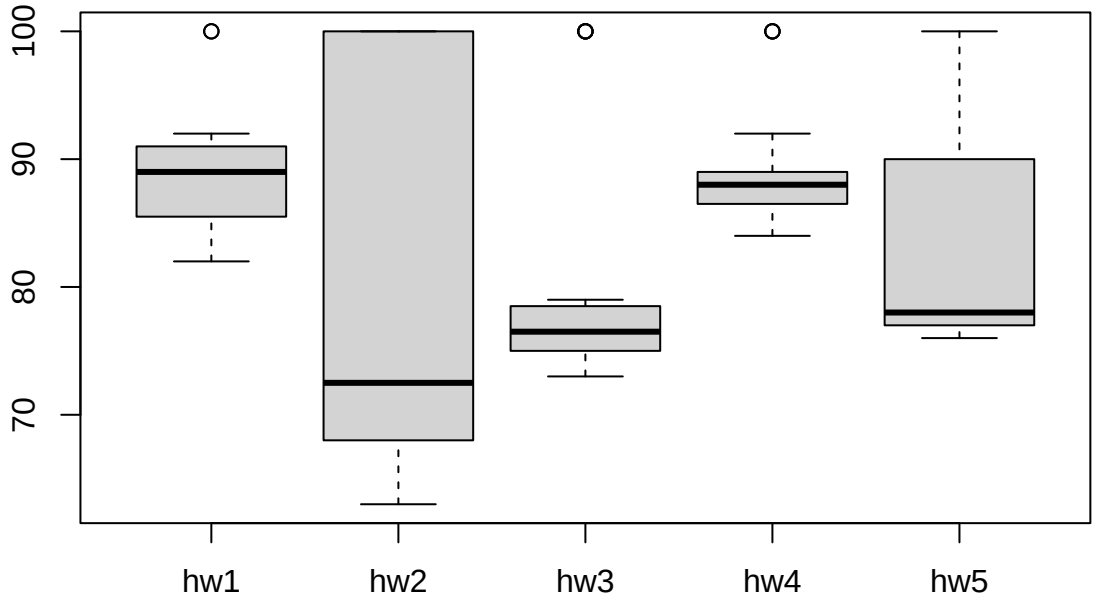
```
med.scores <- apply(gradebook, 2, median, na.rm=TRUE)
med.scores
```

```
##  hw1  hw2  hw3  hw4  hw5
## 89.0 72.5 76.5 88.0 78.0
```

```
which.min(med.scores)
```

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

```
boxplot(gradebook)
```



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]