Class 6: R Functions

Andrew Kapinos (PID: A12708564)

10/15/2021

Quick Rmarkdown Intro

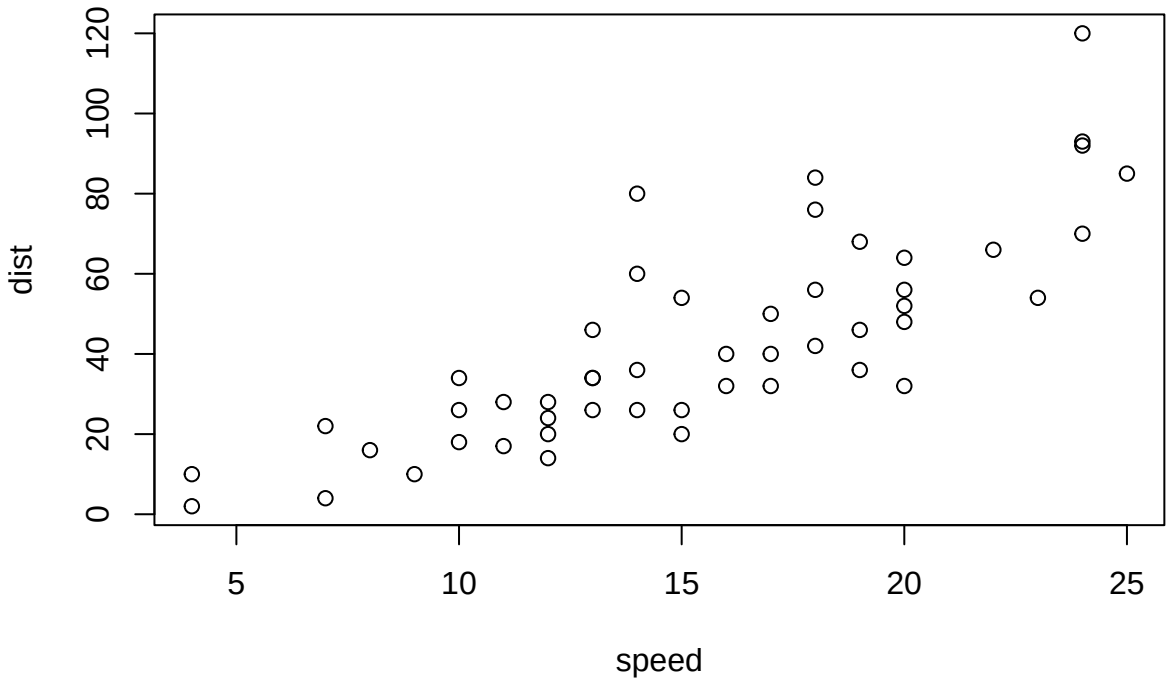
We can write text just like any file. We can **style text to be bold** or *italic*.

Do:

- this
- and that
- and another thing

This is more text
and this is a new line

```
#Comment
plot(cars)
```



Writing Functions

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 adequately 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)
```

We can use **min()** to find the lowest score in each vector.

```
min(student1)
```

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

The **which.min()** function will tell us where in the vector the minimum value is (ie. its position in the vector).

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

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

We can use minus (**"-"**) to get everything in the vector except the lowest score, then calculate the mean of those values using **mean()** (as long as all values are present; ie. no **"NA"** included in vector).

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

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

This does not work for *student2* because the **mean()** function breaks when NA values are present.

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

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

We can identify which values in *student2* are NA using **is.na()**.

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

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

Lets replace NAs with zero, by overriding positions in *student2* containing "NA", using **"="**.

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

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

Now, our original function used on *student1* should work.

```
mean(student.prime[-which.min(student.prime)])
```

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

Let's validate that our approach worked for *student2*.

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

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

Great! Now let's check if our function works for *student3*.

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

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

And now let's validate the result for *student3*.

```
mean(c(90,0,0,0,0,0,0))
```

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

Awesome! Let's simplify and make the function as clear as possible. For instance, we'd like to make the object names concise.

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

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

Let's imagine that the wrong data was entered for one student. In this example, one value was entered as a character string instead of as a numeric string.

```
student4 <- c(100, NA, 90, "90", 90, 90, 97, 80)
```

We can use the **as.numeric()** function to coerce character strings in our vector to be numeric instead.

```
student4
```

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

```
as.numeric(student4)
```

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

Then, we can force the data to be converted to numeric in our first line of code.

```
x <- as.numeric(student4)
x[is.na(x)] = 0
mean(x[-which.min(x)])
```

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

Wonderful! Now we can finally write our function. All functions need 3 things: **a name, input argument(s), and body.**

- We'll name our function **grade**
- Our input argument will be the object **x**, since that will be what's changing between calculations
- The body of the function will be the script we wrote above

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

Let's test our **grade()** function using an example from above, *student1*.

```
grade(student1)
```

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

Now let's grade an entire class

We'll load in our .csv file containing the gradebook information and save it as a data frame. The argument **"row.names=1"** assigns the first column in the .csv file, containing student names, as row titles.

```
gradebook <- "https://tinyurl.com/gradeinput"
scores <- read.csv(gradebook, 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
```

We are going to use the **apply()** function to grade all of the students with our **grade()** function, by applying the function over each row in the data frame.

Required arguments for **apply()** include:

- **x**: the data frame you are applying a function to
- **MARGIN**: a vector giving the subscripts which the function will be applied over; eg., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns
- **FUN**: the function to be applied

```
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
```

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

```
finalgrades <- apply(scores,1,grade)
which.max(finalgrades)
```

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

```
max(finalgrades)
```

```
## [1] 94.5
```

Student 18 is the top scoring student in the gradebook, with an average score of 94.5.

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

We can use the **apply()** function again, but this time looking at the mean scores for each column, as they represent different homework assignments.

```
apply(scores,2,mean)
```

```
##  hw1  hw2  hw3  hw4  hw5
## 89.0   NA 80.8   NA   NA
```

I can ignore the NA values by adding an optional argument to the apply FUN, **na.rm=TRUE**.

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

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

However, this only removes NA values, which may confound the data. Let's instead replace/mask the NA values by overriding positions in *scores* containing "NA", using **"="**.

This is essentially the same process as we performed above, early in Q1.

```
scores2 <- scores
scores2[is.na(scores2)] = 0
scores2
```

```
##            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
```

Now, let's calculate the means for columns in our masked data frame.

```
apply(scores2,2,mean,na.rm=TRUE)
```

```
##   hw1   hw2   hw3   hw4   hw5
## 89.00 72.80 80.80 85.15 79.25
```

Homework 2 (hw2) was the hardest on students.

> 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]

To calculate correlation, we'll use the **cor()** function. The two arguments we'll use are **x** and **y**, using *scores2* and *finalgrades*. Recall that *scores2* contains our masked data frame with "NA"s removed, while *finalgrades* contains students' average scores for all homeworks.

We'll first try using **$** to call a specific homework from *scores2*; in this case, **hw1**.

```
cor(scores2$hw1,finalgrades)
```

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

We can call the **cor()** function for all correlation between average scores and all homework assignments using **apply()**. The way that apply() works, optional arguments for functions go after the function name itself. In this case, the **y** argument thus comes after the col() function name.

```
apply(scores2,2,cor,finalgrades)
```

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

Homework 5 (hw5) has the highest correlation coefficient, and is thus the most predictive of a student's overall score.

Extra: make a boxplot using the score data

```
boxplot(scores)
```