Class 6: R Functions Lab

Elena

2022-10-14

Table of contents

Notes

All functions in R should have at least 3 things:

- A **name** (we pick).
- Input **arguments** (there can be loads - comma-separated).
- A **body** (the R code that does the work).

Useful

- `!` flips the vector e.g., `!c(T,T,F)` gives `c(F,F,T)`.
- `is.na()` returns a vector with F in positions that are not NA, and T in positions that are NA.

- Code > Extract Function to write code into function.

Q1. Writing `grade()` Function

Creating the Code

Problems:

- Identify the lowest single score.
- Drop the lowest single score.
- Determine overall grade.
- Execute function on example gradebook.

```
#Load sample vectors
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)
```

Can use the `mean()` function to find the average

```
#Example for student1
mean(student1)
```

```
[1] 98.75
```

But… we want to drop the lowest grade; can find what the score is using `min()`

```
#Example for student1
min(student1)
```

```
[1] 90
```

```
#Can find more details of function and
#other related functions by looking at help page
?min
```

Can identify the position of the lowest score using `which.min()`.

```r
#Example for student1
which.min(student1)
```

```
[1] 8
```

```r
#Note that student1 does not have any missing assignments (NA)
#student2 and student3 has missing assignments,
#which we will need to take into account later
```

Dropping the lowest score

```r
#Example for student1
student1[-8]
```

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

```r
#Dropping the 8th position, but code may not
#be applicable to other instances

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

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

```r
#More general and helpful
```

Therefore, for student 1, their average grade is:

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

```
[1] 100
```

We need to account for NA values for the other students i.e., making them zero

```r
#Example for student2
mean(student2, na.rm=T)
```

```
[1] 91
```

```
#Note that by doing this, na.rm ignores all the NA values,
#but it is not what we want

#Found `is.na()` function that replaces NA values with specified value
?is.na
```

Breaking down the `is.na()` function

```
#What each element returns
is.na(student2)
```

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

```
#Vector with F for positions that are not NA,
#T for positions that are

student2[is.na(student2)]
```

```
[1] NA
```

```
#Pulls out positions that are NA

student2[is.na(student2)] <- 0
#Replaces NA with specified value

student2
```

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

Putting the steps together…

```
#Example for student2
student2[is.na(student2)] <- 0
mean(student2[-which.min(student2)])
```

```
[1] 91
```

Writing the Function!

Simplifying the code…

```
#Example for student1

x <- student1

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

```
[1] 100
```

Now to make the function

- Instead of typing it out, can highlight the code and click Code > Extract Function and R will format accordingly

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

#x is input argument
#Remember to load this function before using!
```

Testing the function

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Q2. Applying Function to Gradebook & Identifying Top Student

Reading CSV file

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

#Note that the students column is part of the data;
#we want to make it into rownames

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

```
          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
```

Now want to introduce the `apply()` function

```
?apply
#Similar to a for loop, where it applies the function to each
#Syntax is apply(X,MARGIN,FUN)
#X is input i.e., gradebook
#MARGIN 1=rows, 2=columns
#FUN is function


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

```
 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
```

Can use `which.max()` to find where the largest/max value is in this results vector. Therefore, the top-scoring student is:

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

```
student-18
        18
```

```r
#To find their score, can use `max()`
max(results)
```

```
[1] 94.5
```

Q3. Toughest Homework

Finding the average of the homework assignments

```r
#We can use `apply()` again,
#but this time over the columns i.e., MARGIN = 2
#Remember we cannot use grade function because
#it will get rid of the lowest scoring homework in each column
#Use sum instead, taking into account NA using `na.rm=T`

homework <- apply(gradebook, 2, sum, na.rm=T)
homework
```

```
 hw1  hw2  hw3  hw4  hw5 
1780 1456 1616 1703 1585 
```

The toughest homework i.e., homework with the lowest score is:

```r
which.min(homework)
```

```
hw2 
  2 
```

Q4. Homework Most Predictive of Overall Score

Performing a Pearson correlation

```
?cor
cor(gradebook$hw5,results)
```

```
[1] NA
```

```
#Need to fix NA values first
```

Making NA values 0

```
mask <- gradebook
mask[is.na(mask)] <- 0
head(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
```

Now performing Pearson correlation

```
hw <- apply(mask, 2, cor, results)
hw
```

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

The homework with the most predictive score is therefore:

```
which.max(hw)
```

```
hw5 
  5 
```

Q5. Render Document