Class06

AUTHOR
Nichelle Camden

PUBLISHED
October 14, 2022

Function basics

All functions in R have at least 3 things:

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

```
# 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)

#and later we'll try it with this data
gradebook <- read.csv("student_homework.csv")
```

I can use the `mean()` function to get the average

```
mean(student1)
```

```
[1] 98.75
```

To find the lowest value, I can use the `min()` function

```
student1
```

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

```
min(student1)
```

```
[1] 90
```

I found the `which.min()` function, what does it do?

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

```
[1] 8
```

The minimum value is in the 8th position for “student1”.

There are a few different ways to take that value out from the average calculation, but the index trick might be the easiest/most useful. (It could also work to sort the students' scores from lowest to highest, and then drop the first score for all.)

```
#lil refresh on the index trick:
student1[-8]
```

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

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

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

Then I can take the mean

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

```
[1] 100
```

Will it work with "student2"?

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

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

Kind of, but not really. It just got rid of the lowest numerical value, but not the actual lowest value (the NA)

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

```
[1] NA
```

```
mean(student2, na.rm=T)
```

```
[1] 91
```

We need another way…

Can I replace NA values with zero? No homework submission = 0 try google

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

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

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

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

```r
c(T,T,F)
```

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

```r
!c(T,T,F)
```

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

```r
# the ! flips it
```

```r
is.na(student3)
```

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

```r
student3[ is.na(student3) ] <- 0
student3
```

```
[1] 90  0  0  0  0  0  0  0
```

```r
positions <- is.na(student2)
student2
```

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

```r
student3[positions] <- 0
student3
```

```
[1] 90  0  0  0  0  0  0  0
```

```r
#could also use "missing" instead of "positions"
```

```r
student2[is.na(student2)] <- 0
mean(student2 [-which.min(student2)])
```

```
[1] 91
```

Re-write my snippet to be more simple for **Q1**

Q1

```r
x <- student1
x[is.na(x)] <- 0
```

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

```
[1] 100
```

```
#can replace "x" with student2 and 3
x <- student2
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
[1] 91
```

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

```
[1] 12.85714
```

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

Now use that to grade student1 and others

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Another way to do above is to highlight snippet -> Code (at top) -> Extract Function -> name the function (here I named it “grade”)

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

Q2

Who is the top scoring student overall in the gradebook?

(can get the data this way or the way I used above with the “read.csv(”file_name”) if it’s downloaded already)

```
gradebook <- read.csv("https://tinyurl.com/gradeinput",
row.name = 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 I want to introduce the `apply()` 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 
```

I can use `which.max` to find where the largest/ max value is in this results vector

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

```
student-18 
        18 
```

Q3

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

We can use `apply()` again, but this time over columns (use 2 instead of 1 so margin=2)

```
apply(gradebook, 2, sum, na.rm= TRUE)
```

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

```
lowest_score <- apply(gradebook, 2, sum, na.rm= TRUE)
lowest_score
```

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

I can use my eyeballs to see that homework 2 was the toughest, but I can also get R to tell me explicitly (incase datasets are too big in the future)

```
which.min(lowest_score)
```

```
hw2 
  2 
```

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)?

```
#cor(gradebook$hw1, results)
#cor(gradebook$hw2, results)

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

```r
cor(mask$hw5, results)
```

```
[1] 0.6325982
```

It looks like homework 5 is highly correlated, but let's use the `apply()` function over the masked gradebook so we don't have to retype hw1, hw2, etc

```r
apply(mask, 2, cor, y=results)
```

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