class06

Laurie Chang A16891192

2024-01-25

R Functions

Functions are how we get stuff done. We call functions to do everything useful in R. One cool thing about R is that it makes writing your own functions comparatively easy.

All functions in R have at least three things:

1. a **name** (we get to pick this)
2. one or more **input arguments** (the input to our function)
3. the **body** (lines of code that do the work)

```
funname <- function(input1, input2) {
  # the body with R code
}
```

Let's write a silly first function to add 2 numbers:

```
x <- 5
y <- 1
x + y
```

```
[1] 6
```

```
addme <- function(x,y) {
  x + y
}
```

```
addme(1,1)
```

```
[1] 2
```

```
addme <- function(x,y = 1) {
  x + y
}
```

```
addme(10)
```

```
[1] 11
```

Lab for today

Write a function to grade student work from class.

Start with a simplified version of the problem.

```
# 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)
```

Let's just find the average.

```
mean(student1)
```

```
[1] 98.75
```

```
mean(student2)
```

```
[1] NA
```

Student 2 gives a NA, why? NA is in the sequence!

Let's try using the `na.rm` argument in order to get rid of the NAs in the sequence.

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

```
[1] 91
```

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

```
[1] 90
```

The third student only had one assignment turned in but their average was a 90; is that fair? No! Student 3 should not have a mean of 90!

This argument is not useful in this case as we want to only remove the lowest score they have. This argument removes all the NAs (even if there is more than 1).

Come back to this NA problem. But things worked for `student1`.

We want to drop the lowest score bfore getting the `mean()`.

How do I find the lowest (minimum) score?

```
min(student1)
```

```
[1] 90
```

It does return the lowest score but we want to remove this. The `min()` function doesn't work for student 2 or 3 anyways.

I found the `which.min()` function. Maybe this is more useful?

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

```
[1] 8
```

An 8 is returned as the lowest score (90) is in the 8th element of the vector. This can be rewritten as:

```
# Find the lowest score
student1[which.min(student1)]
```

```
[1] 90
```

How do you return everything BUT this one value?

Add a minus to get everything BUT that value!

```
# Remove the lowest score
student1[-which.min(student1)]
```

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

Now find the mean without that 8th element to get the average with the dropped score!

```
#Find the mean without the lowest score
mean(student1[-which.min(student1)])
```

```
[1] 100
```

You can assign `which.min()` to a value in order to reduce typing, but it's not necessary.

Use a common shortcut and use `x` as my input.

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

```
[1] 100
```

If I do this for student 2, will it work?

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

```
[1] NA
```

No; I still get NA.

We still have the problem of missing values.

One idea is to replace NA values with 0.

We have made something equal another value before, but NAs are a little different.

```
y <- 1:5
y[y==3] <- 10000
y
```

```
[1]     1     2 10000     4     5
```

```
y <- c(1, 2, NA, 3, 4, 5)
y == NA
```

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

This does not work.

Use the `is.na()` function to replace

```
y
```

```
[1]  1  2 NA  3  4  5
```

```
is.na(y)
```

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

How can I remove the NA elements from the vector?

```
#y[is.na(y)]
# putting an exclamation point in front of the vector flips it
!c(F,F,F)
```

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

```
y[!is.na(y)]
```

```
[1] 1 2 3 4 5
```

```
x <- student1

#change NA values to 0
x[ is.na(x)] <- 0
#find and remove min value and get mean
mean(x[-which.min(x)])
```

```
[1] 100
```

```
x <- student2

#change NA values to 0
x[ is.na(x)] <- 0
#find and remove min value and get mean
mean(x[-which.min(x)])
```

```
[1] 91
```

```
x <- student3

#change NA values to 0
x[ is.na(x)] <- 0
#find and remove min value and get mean
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

Last step now that I have my working code snippet is to make my `grade()` function.

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

Test out the function!

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Now read the online gradebook file (CSV file)

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

rename the column of the first column to be the names of the students

```
gradebook <- read.csv(url, 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
```

Can use the ? function to help understand the function. If you don't understand, can also use claude ai to get the explanation.

Use what we learned to print out our results!

```
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
```

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

```
max(results)
```

```
[1] 94.5
```

```
  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?

```
  grade2 <- function(x) {
    x[ is.na(x)] <- 0
    mean(x)
  }
  apply(gradebook, 2, grade2)
```

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

```
  which.min(apply(gradebook, 2, grade2))
```

```
hw2
  2
```

you could also use the `mean()` function IF you add some more arguments! Applying arguments after the function will apply the arguments to that function.

```
  apply(gradebook, 2, mean, na.rm = T)
```

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

```
  which.min(apply(gradebook, 2, mean, na.rm = T))
```

```
hw3
  3
```

Mean is super sensitive to outliers/extreme values. Let's use the sum function to see total summed scores.

```
which.min(apply(gradebook, 2, sum, na.rm = T))
```

```
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)?

another way (making NA = 0)

```
#make all NA (on mask) 0
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
```

we can use the `cor()` function for correlation analyses

```
#does the score the student got for hw1 correlate with their results?
cor(gradebook$hw1, results)
```

```
[1] 0.4250204
```

mid correlation; what about the others?

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

```
[1] 0.6325982
```

```
cor(mask$hw4, results)
```

```
[1] 0.3810884
```

```
cor(mask$hw3, results)
```

```
[1] 0.3042561
```

```
cor(mask$hw2, results)
```

```
[1] 0.176778
```

Above 0.6 is high correlation!

Need to use the `apply()` function to run this analysis over the whole course (ie. masked gradebook)

```
#apply(mask, 2, cor)
#this does not work as cor needs an x and a y; add the argument of y after cor (results) w

apply(mask, 2, cor, results)
```

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