Class 6: R Functions

AUTHOR
Nundini Varshney (PID: A16867985)

PUBLISHED
January 25, 2024

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:

- A **name** (we get to pick this)
- One or more **input arguments** (the input to our function)
- The **body** (lines of code that do the work)

```
#/ eval: false

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

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

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

```
[1] 6
```

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

```
addme(100,100)
```

```
[1] 200
```

```
addme(10)
```

```
[1] 11
```

Lab for today

Write a function to grade student work from class.

Start with a simplified version of the problem:

```r
# 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 find the average.

```r
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

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

```
[1] 90
```

This is not fair - there is no way student3 should have a mean of 90.

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

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

How do I find the lowest (minimum) score?

```r
student1
```

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

```r
min(student1)
```

```
[1] 90
```

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

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

```
[1] 8
```

Cool - it is the 8th element of the vector that has the lowest score. Can I remove this one?

```
student1[ 8 ]
```

```
[1] 90
```

We can use the wee minus trick for indexing.

```
x <- 1:5
x[-3]
```

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

Now put these bits of knowledge together to make some code that identifies and drops the lowest score (element of the input vector) and then calculates the mean.

```
# Find the lowest score
ind <- which.min(student1)
# remove lowest score and find the mean
mean( student1[-ind])
```

```
[1] 100
```

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

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

```
[1] 100
```

We still have the problem of missing values.

One idea is to replace NA values with zero.

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

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

Bummer, this is no good…

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

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

```
y
```

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

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

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

How can I remove the NA elements from the vector?

```
!c(F,F,F)
```

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

```
#y[-is.na(y)]
```

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

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

```
y [ is.na(y) ] <- 10000
y
```

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

OK lets solve this:

```
x <- student3

# Change NA values to Zero
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.

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]

```
grade <- function(x) {
  x[ is.na(x) ] <- 0
  # Find and remove min value and get mean
  mean( x[ -which.min(x) ] )
}
```

```r
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
grade(student3)
```

```
[1] 12.85714
```

Now read the online gradebook (CSV file)

```r
url <- "https://tinyurl.com/gradeinput"
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
```

```r
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? [3pts]

```r
max(results)
```

```
[1] 94.5
```

```r
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? [2pts]

```
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 
```

```
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)? [1pt]

```
# Make all (or mask) NA to zero
mask <- gradebook
mask[ is.na(mask) ] <- 0
#mask
```

We can use the `cor()` function for correlation analysis.

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

```
[1] 0.6325982
```

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

```
[1] 0.3042561
```

I need to use the `apply()` function to run this analysis over the whole course (i.e. masked gradebook)

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

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

Q5. Make sure you save your Quarto document and can click the “Render” (or Rmarkdown”Knit”) button to generate a PDF format report without errors. Finally, submit your PDF to gradescope. [1pt]