Class 6: R Functions

Richard Gao (PID: A16490010)

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:

- 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)

```
funname <- function(input 1, input 2) {
  # 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(1, 1)
```

```
[1] 2
```

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

```
[1] 200
```

```
addme(10)
```

```
[1] 11
```

Lab for Today

```
# 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)
```

Test functions.

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

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

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

```
[1] 8
```

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

```
[1] 90
```

```
cor(student2, student3)
```

```
[1] NA
```

Test na.rm.

```
  mean(student1, na.rm = TRUE)
```

```
[1] 98.75
```

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

```
[1] 91
```

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

```
[1] 90
```

This is not fair - there is no way student3 should have a mean of 90. We want to drop the lowest score before getting the `mean()`

How do I find the lowest score?

```
  min(student1)
```

```
[1] 90
```

But I need the location of the lowest score.

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

```
[1] 8
```

Cool - it is the eighth position but how do I remove that from the vector of grades?

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

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

Now find the mean of the dropped vector.

```
ind <- which.min(student1)
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. Replace NA values with 0.

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

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

```
student3[is.na(student3)] = 0
student3
```

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

How can I remove the NA elements from the vector?

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

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

```
y <- c(1, 2, NA, 4, 5)
y[!is.na(y)]
```

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

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"

```
grade <- function(student) {
  # Change NA values to Zero
  student[is.na(student)] = 0
  # Find and drops min value and calculates mean
  mean(student[-which.min(student)])
}
```

Test the grade function.

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Now read the online gradebook(CSV file)

```
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 apply the grade function to the CSV gradebook.

```
  # 1 indicates rows and 2 indicates columns
  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
```

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

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

```
student-18
        18
```

```
  max(results)
```

```
[1] 94.5
```

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

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

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

```
  which.min(AvgHW)
```

```
hw3
  3
```

Different because one student in HW3 had a very low score that pulled the HW3 average down.

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

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

```
which.min(AvgHW)
```

```
hw2
  2
```

Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)?

```
# Make all NA values 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 analysis.

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

```
[1] 0.6325982
```

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

```
[1] 0.3042561
```

Try to apply to gradebook:

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

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