Class 6: R functions

Neha (PID: A17567541)

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 3 things:

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

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

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

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

```
[1] 6
```

```
addme<-function(x,y=100) {x+y}
```

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

```
[1] 2
```

```
addme(10)
```

```
[1] 110
```

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)
```

```
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

Is there a way to set the value of NA to 0?.

Come back to the NA problem. We want to drop the lowest score before getting `mean()`

How do I find the lowest (minimum) score?

```
min(student1)
```

```
[1] 90
```

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

```
#Tells you which element has the minumum
which.min(student1)
```

```
[1] 8
```

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

```
#Find the lowest score and remove it
student1[-which.min(student1)]
```

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

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

```
[1] 100
```

[] is used for which values you are calling out of a vector

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 0

This does not work.

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

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

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

How can I remove the NA element from the vector. Use ! to flip logicals.

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

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

```
y[is.na(y)]<-1000
```

```
x<-student3

x[is.na(x)]<-0
#Changes the NA values to 0
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

```
#Gives the mean
```

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

grade()<-function(x,y=1) {x[is.na(x)]<-0 #Changes the NA values to 0 mean(x[-which.min(x)]) #Gives the mean}

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

```
[1] 100
```

Now read the online gradebook

```
url<-"https://tinyurl.com/gradeinput"
gradebook<-read.csv(url, row.names=1)
#row.names function sets the names of the rows. In this case, row.names 1 will set it to c

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
```

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

```
  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? [2pts]

```
  grade1<-function(x) {
    x[is.na(x)]<-0
  mean(x)
  }
  #If we used the original grade function, it would subtract the lowest scores, which would
  results2<-apply(gradebook,2,grade1)
  results2
```

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

```
  min(results2)
```

```
[1] 72.8
```

```
  which.min(results2)
```

```
hw2
  2
```

Or, we could add an additional argument to the `apply()` function

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

```
hw3
  3
```

```
  #na.rm removes the NA values
```

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

```
hw2
  2
```

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

```
[1] 1456
```

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

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

Using the mean might give the wrong impression, because outliers may affect the data significantly. If we omit the NA values and compute the sum, we will get a better measure of how students did.

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 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$hw2, results)
```

```
[1] 0.176778
```

```
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)

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

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