

Class 6: R functions

Lilia Jimenez (PID:A16262599)

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 choose this) -1 or more **Input arguments** (the inputs to our function) -The **body** (lines of code that do the work)

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

lets 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(100)
```

```
[1] 101
```

Lab for today

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)
```

lets just find the average

```
mean(student1)
```

```
[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!

come back to this NA prob. but things worked for student1

we want to drop 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?

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

```
[1] 8
```

cool this is the 8th element of the vector that has the lowest score. how can i remove this one?

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

```
[1] 90
```

we can use the wee inus trick for indexing

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

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

```
#find the lowest score  
ind<- which.min(student1)  
#remove the lowest score  
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 problems 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
```

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

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

How can I remove the NA element from the vector?

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

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

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

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

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

```
[1] 1 2 0 4 5
```

Ok lets put the pieces together

```
x<-student3

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

```

Now read the online gradebook (CSV file)

```

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

```

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

```

max(results)

```

[1] 94.5

```

which.max(results)

```

```
student-18
      18
```

q3.

```
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

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

We can use the ‘cor()’ function for correlation analysez

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

```
[1] 0.6325982
```

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

```
[1] 0.3042561
```

I need to use the ‘apply()’ function to run this analysez over the whole course

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

hw1	hw2	hw3	hw4	hw5
0.4250204	0.1767780	0.3042561	0.3810884	0.6325982