Class06

Anna Waters (PID: A16271985)

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 or accessible.

All functions in R have at least 3 things:

- A **Name** (we get to pick this)
- **Input Arguments** (the input to our function, one or more)
- The **body** (lines of code that do the work)

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

Lets write a silly first function

```
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 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.

```
# 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)
```

Snippet for the answers to the question on student 1

```
mean(student1) #average grade for student
```

```
[1] 98.75
```

```
mean(student2,na.rm = TRUE) # issue with NA
```

```
[1] 91
```

```
mean(student3,na.rm=TRUE) # can't just remove all NA's bc this isnt right
```

```
[1] 90
```

Come back to the NA problem.

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

```
#tells the lowest score but not the location
min(student1)
```

```
[1] 90
```

I found `which.min()` function. Maybe this is more useful? It should give the location of the lowest score

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

```
[1] 90
```

We can use the little minus trick from indexing

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

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

Using the index trick to produce a vector without the lowest value and using `mean()` on the new vector to find the average without the lowest value.

```
# vector w/o the lowest score
student1[-which.min(student1)]
```

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

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

```
[1] 100
```

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

```
x <- student3
mean(x[-which.min(x)])
```

```
[1] NA
```

We still have the problem of missing values. One idea is to replace NA values with 0.

```
y <- 1:5
#replacing 3 with 1000 by seeing which = 3 & replacing
y[y == 3] <- 1000
y
```

```
[1]    1    2 1000    4    5
```

Bummer, NA's are special so == does not work

```
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 elements from the vector? First I need to flip the TRUE elements to FALSE

```
#! flips logicals
!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)] <- 1000
```

Okay lets put humpty dumpty back together again.

```
x <- student3

#assigning where NA = TRUE, the value of 0/ change NA 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 a working code snippet is to make my `grade()` function

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

Testing `grade()` function on the test students

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Read in the online gradebook

```
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
```

Applying the `grade()` function to the sample data from the csv.

```
# apply function uses the function on each row (margin 1) instead of needing a for loop to
results <- apply(gradebook,MARGIN = 1,FUN = 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?

```
#which.max locates the highest scoring student
which.max(results)
```

```
student-18
        18
```

```r
#what the max average value is
max(results)
```

```
[1] 94.5
```

Looking above, the top student is student 18.

Q3. From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall? [2pts]

```r
# apply the grade function by column to find averages per hw
hw_results <- apply(gradebook,MARGIN = 2,mean,na.rm=T)

#which homework got the lowest result
which.min(hw_results)
```

```
hw3
  3
```

```r
#lost score of the hw
min(hw_results)
```

```
[1] 80.8
```

Mean is susceptible to outliers so `sum()` can be used instead. This combats the skew.

```r
hw_sum <- apply(gradebook,MARGIN = 2,sum,na.rm=T)

#which homework got the lowest result
which.min(hw_sum)
```

```
hw2
  2
```

```r
#lost score of the hw
min(hw_sum)
```

```
[1] 1456
```

Looking above, the toughest homework, taking into account skew is hw2.

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 NA equal to zero in mask
mask <- gradebook
mask[is.na(mask)] <- 0
#mask
```

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

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

```
[1] 0.6325982
```

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

```
[1] 0.3042561
```

```
apply(mask, MARGIN = 2, FUN = cor, results)
```

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

Looking above, hw5 is the most correlated with student results.

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