class06: R Functions

Chen(PID=A59026768)

2024-01-26

The first silly function

All functions in r have three parts, that are:

- a name
- input arguments (non, one, or more)
- a body

A function to add two numbers

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

Let me try out this function:

```
sillyadd(10)
```

```
[1] 11
```

Let's do something more useful

```
student1 <-c(100,100,100,100,100,100,100,90)
student2 <-c(100, NA, 90, 90, 90, 90, 97, 80)

sst1 <- sort(student2)
dsst1 <- sst1[-1]
mean(dsst1)
```

```
[1] 92.83333
```

```
student1 <-c(100,100,100,100,100,100,100,90)
# Find lowst value
lowest <- which.min(student1)
lowest
```

```
[1] 8
```

```
dstudent <- student1[-lowest]
dstudent
```

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

```
mean(dstudent)
```

```
[1] 100
```

```
student2 <-c(100, NA, 90, 90, 90, 90, 97, 80)
NaV <- is.na(student2)
student2[NaV] <- 0
student2
```

```
[1] 100   0  90  90  90  90  97  80
```

```
lowest <- which.min(student2)
lowest
```

```
[1] 2
```

```
dstudent <- student1[-lowest]
dstudent
```

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

```
mean(dstudent)
```

```
[1] 98.57143
```

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"

```
grade <- function(x, drop.lowest=TRUE){
  x[is.na(x)] <- 0
  if(drop.lowest){
    ox <- sort(x)
    dox <- ox[-1]
    mdox <- mean(dox)
  }
  else {
    mdox <- mean(x)
  }
  mdox

}
```

```
grade (student1)
```

```
[1] 100
```

```
url <- "https://tinyurl.com/gradeinput"
gradebook <- read.csv(url, row.names = 1)
```

Because gradebook is a data.frame instead of a vector, the function grade() can't work. We can use function apply() to apply grade() to all the dataframe instead of using for/while loop.

Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

```
FF<-apply(gradebook, MARGIN = 1, grade)
F <-which.max(apply(gradebook, MARGIN = 1, grade))
F
```

```
student-18
        18
```

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

```
HW <- apply(gradebook, MARGIN = 2, grade)
HW
```

```
     hw1      hw2      hw3      hw4      hw5
89.36842 76.63158 81.21053 89.63158 83.42105
```

```
THW <-which.min(HW)
THW
```

```
hw2
  2
```

```
allhw <- apply(gradebook, MARGIN = 2, grade, drop.lowest=F)
```

Q4Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? [1pt]

```
mask <- gradebook
mask[is.na(mask)]<-0
mask$hw5
```

```
 [1]  79  78  77  76  79  77 100 100  77  76 100 100  80  76   0  77  78 100  79
[20]  76
```

```
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
```

```
FF
```

```
 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
```

```
which.max(cor(mask, FF))
```

```
[1] 5
```