Class 6: R functions

Tianru Zhang (PID: A15432834)

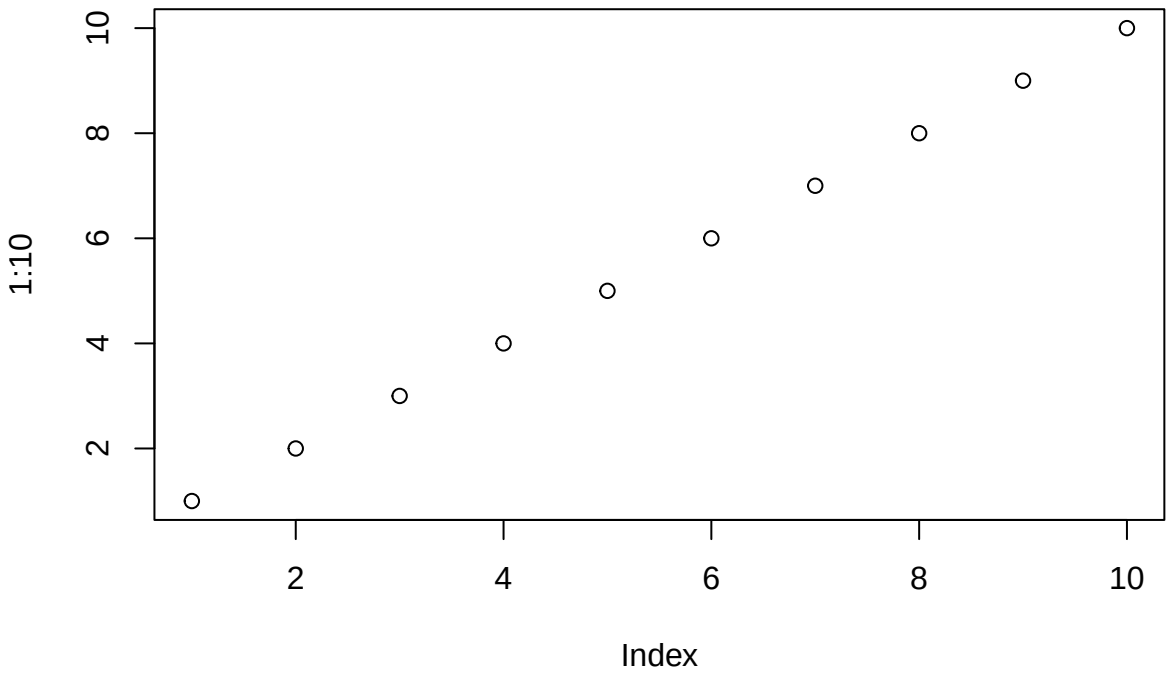
10/14/2021

##Playing with R-markdown

In R markdown I can **BOLD** things and *Italic* things

```
#This is a code chunk so text in this area needs #to be commented

plot(1:10)
```



##R functions In today's text I am about to write some functions that grades students' homework.

Questions for today:

> **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]

```
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)
```

Let's start with student1 and find their mean score

```
mean(student1)
```

```
## [1] 98.75
```

Let's find the lowest score

```
min(student1)
```

```
## [1] 90
```

The **which .min()** function seems helpful here:

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

```
## [1] 8
```

This function gives the position of the lowest score!

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

```
## [1] 8
```

```
# this is the lowest score of the student1
student1[which.min(student1)]
```

```
## [1] 90
```

TO drop this min value, can do the minus

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

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

We are so close now to find the mean of the student's score

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

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

Now I am trying whether it works with student2

```
mean(student2[-which.min(student2)], na.rm=TRUE)
```

```
## [1] 92.83333
```

This is not what we wanted, because it dropped the lowest score 80, and not the NA

Look at student3:

```
mean(student3[-which.min(student3)], na.rm=TRUE)
```

```
## [1] NaN
```

One idea: replace all the NA with zeros Try student 2

```
is.na(student2)
```

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

This **is.na** function returns the logical vector where the TRUE elements represent the NA values' positions in the vector

```
which(is.na(student2))
```

```
## [1] 2
```

Let's replace the NA values with zeros.

```
# make a copy of student2's results
student.prime<-student2
student.prime
```

```
## [1] 100  NA  90  90  90  90  97  80
```

```
student.prime[which(is.na(student2))]=0
mean(student.prime[-which.min(student.prime)])
```

```
## [1] 91
```

```
mean(c(100,90,90,90,90,97,80))
```

```
## [1] 91
```

Looks like this method works!!

Check student3: This time, with a simplified method

```
x<-student3
#Map NA values to zero
x[which(is.na(student3))]=0
#find the mean after removal of the lowest value
mean(x[-which.min(x)])
```

```
## [1] 12.85714
```

Now we need to put this all together to get the average score dropping the lowest or where the NAs are; get the body of the function

```
grade<- function(y){
  #Make sure the score inputs are all numbers
  y<-as.numeric(y)

  #Map NA values to zero
  y[which(is.na(y))]=0
  #find the mean after removal of the lowest value
  mean(y[-which.min(y)])
}
```

```
grade(student1)
```

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

```
grade(student2)
```

```
## [1] 91
```

```
grade(student3)
```

```
## [1] 12.85714
```

It seems that the function works!

Now read in the csv file

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

```
##            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
## 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  NA  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  NA
## student-16  92 100  74  89  77
```

```
## student-17  88  63 100  86  78
## student-18  91  NA 100  87 100
## student-19  91  68  75  86  79
## student-20  91  68  76  88  76
```

Now try the function on students:

First, explore the as.numeric() function:

```
grade(as.numeric(scores[2,]))
```

```
## [1] 82.5
```

```
as.numeric(c(1,2,NA,4,5))
```

```
## [1]  1  2 NA  4  5
```

There are some non-numeric values in the csv file. as.numeric would turn a row into numeric vector.

Now grade all students using the **apply()** function.

```
ans<-apply(scores,1,grade)
ans
```

```
##  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]

```
which.max(ans)
```

```
## student-18
##         18
```

As is seen from the results above, the highest score is **94.50**, from **student-18**

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

We can use the apply() function to set the margin=2 argument:

```
apply(scores,2,mean,na.rm=TRUE)
```

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

As is seen from the results above, the **hw3** has the lowest mean score, of **80.08**

> Q4.

```
mask<-scores
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
```

```
#cor(ans,mask$hw3)
apply(mask,2,cor,ans)
```

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

As can be seen above, hw5 is the most predictive of the overall scores.