Class 6: R Functions

Jaquish (PID: A59010386)

10/15/2021

Quick Rmarkdown intro

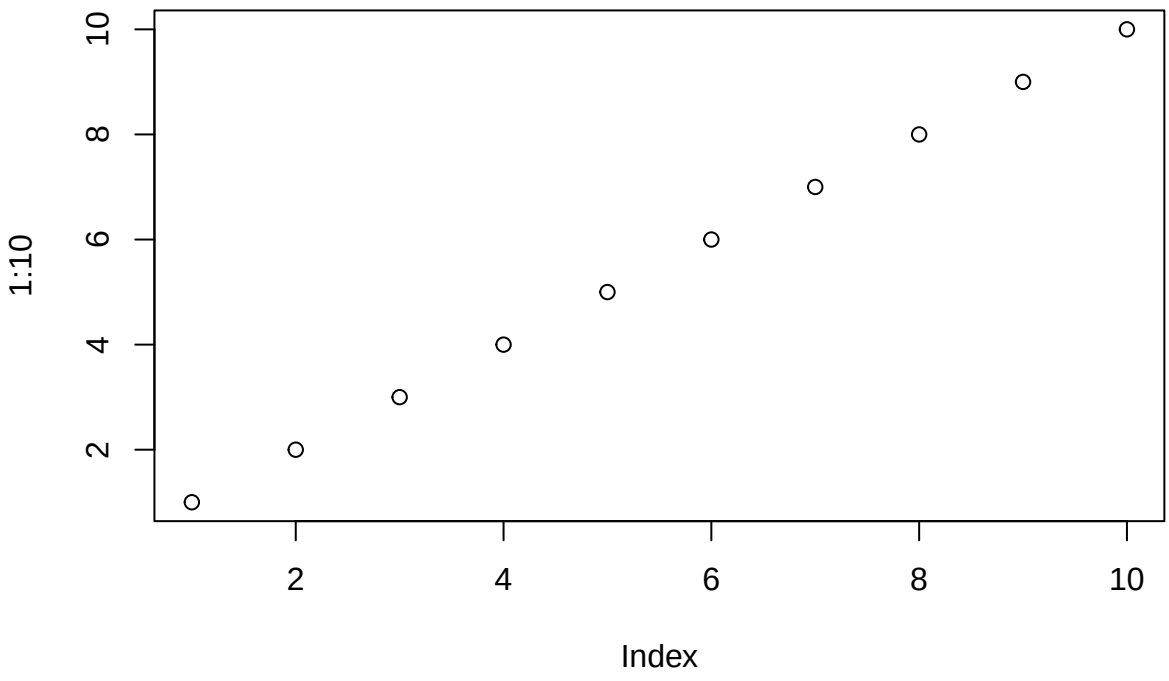
We can write text of course just like any file. **We can style text to be bold** or *italic*.

Do:

- this
- and that
- and another thing

We can include some code:

```
plot(1:10)
```



Shortcut for inputting new r code = OPTION+COMMAND+I

Time to write a function

> **Q1** Q1. Write a function grade() to determine an overall grade from a vector of student homeworkassignment 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]

```
# 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)
```

Student1

First I want to find the lowest score, can use the **min()** function. Can then use the **which.min()** function to determine where the lowest is (it's position in the vector).

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

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

Can use the square bracket with a minus - to remove the lowest score in the vector.

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

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

Then use the mean function with the above to find the average of the remaining values in the vector.

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

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

Student2

```
mean(student2[-which.min(student2)])
```

```
## [1] NA
```

NO. . . there is an NA in the vector so this same code will not work.

```
student2
```

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

```
which.min(student2)
```

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

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

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

One great idea is to replace the na values with 0.

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

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

This is.na() function returns a logical vector where TRUE indicates the presence of na.

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

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

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

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

```
student2[is.na(student2)]
```

```
## [1] NA
```

Lets replace NA's with 0.

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

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

Student3

```
student.prime3=student3
student.prime3[is.na(student3)]=0
mean(student.prime3[-which.min(student.prime3)])
```

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

Great ! Let's simplify.

We can make the variable/group names easier.

```
x=student3
x[is.na(student3)]=0
mean(x[-which.min(x)])
```

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

IF the information was entered wrong...

```
student4=c(100, NA, 90, "90", 90, 90, 97, 80)
```

```
x=student4
x=as.numeric(x)
x[is.na(x)]=0
mean(x[-which.min(x)])
```

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

Now finally we can write our function: All functions have at least 3 things. A name, input args and a body.

```
grade=function(x) {
  x=as.numeric(x)
x[is.na(x)]=0
mean(x[-which.min(x)])
}
```

```
grade(student1)
```

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

##Now grade a whole class

First we got to read the gradebook for the class.

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

```
gradebook="https://tinyurl.com/gradeinput"
scores=read.csv(gradebook, 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
```

```
ans=apply(scores,1,grade)
```

ans=apply(scores,1,grade)

We are going to use the super useful **apply()** function to grade all the students with out **grade()** function

```
apply(scores, 1, grade)
```

```
##  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
```

Because we are using the apply function indicating #1 means we are applying that function to every row (horizonatl) giving us the average value for every student. IF we did ,2, we would get the average for every column or in this case each hw.

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

```
which.max(apply(scores, 1, grade))
```

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

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

```
apply(scores,2,grade)
```

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

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

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

```
ScoreHW=function(x) {
  x=as.numeric(x)
x[is.na(x)]=0
mean(x)
}
apply(scores,2,ScoreHW)
```

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

```
which.min(apply(scores,2,ScoreHW))
```

```
## hw2
##   2
```

Replace or mask NA values to zero.

```
mask=scores
is.na(mask)
```

```
##              hw1   hw2   hw3   hw4   hw5
## student-1  FALSE FALSE FALSE FALSE FALSE
## student-2  FALSE FALSE FALSE FALSE FALSE
## student-3  FALSE FALSE FALSE FALSE FALSE
## student-4  FALSE  TRUE FALSE FALSE FALSE
## student-5  FALSE FALSE FALSE FALSE FALSE
## student-6  FALSE FALSE FALSE FALSE FALSE
## student-7  FALSE FALSE FALSE FALSE FALSE
## student-8  FALSE FALSE FALSE FALSE FALSE
## student-9  FALSE FALSE FALSE FALSE FALSE
## student-10 FALSE FALSE FALSE  TRUE FALSE
## student-11 FALSE FALSE FALSE FALSE FALSE
## student-12 FALSE FALSE FALSE FALSE FALSE
## student-13 FALSE FALSE FALSE FALSE FALSE
## student-14 FALSE FALSE FALSE FALSE FALSE
## student-15 FALSE FALSE FALSE FALSE  TRUE
## student-16 FALSE FALSE FALSE FALSE FALSE
## student-17 FALSE FALSE FALSE FALSE FALSE
## student-18 FALSE  TRUE FALSE FALSE FALSE
## student-19 FALSE FALSE FALSE FALSE FALSE
## student-20 FALSE FALSE FALSE FALSE FALSE
```

```
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
```

Now we can use apply on our "masked" scores

```
apply(mask,2,mean)
```

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

```
which.min(apply(mask,2,mean))
```

```
## hw2
##   2
```

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]

Here we will use the **cor()**

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

```
## [1] 0.6325982
```

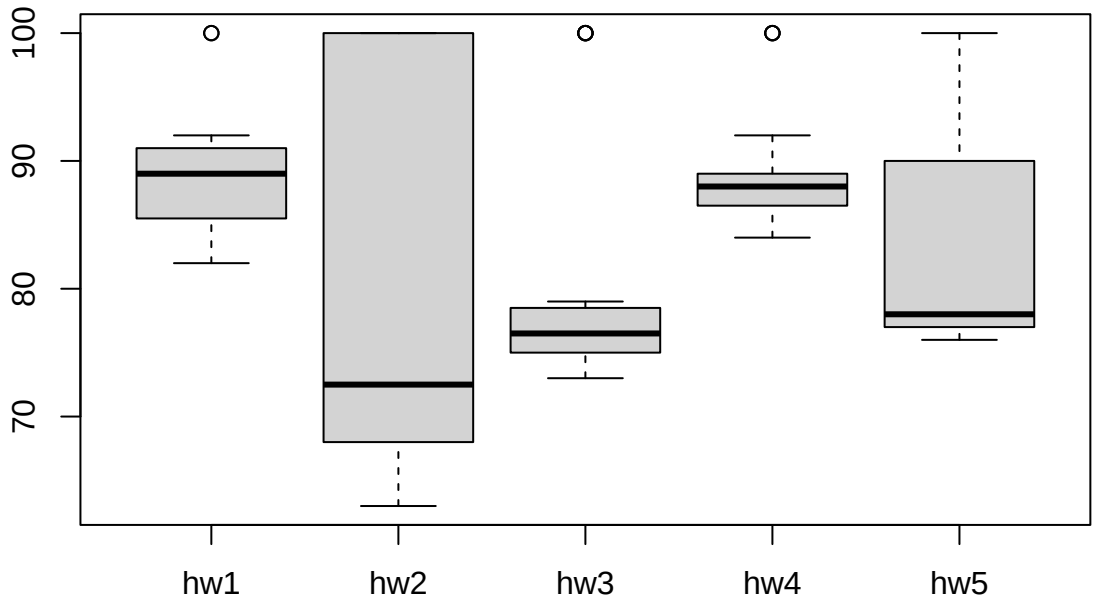
I can call the **cor()** for every hw and get a value for each but that sucks, lets use **apply()** and do them all in one go.

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

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

Make a boxplot

```
boxplot(scores)
```




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

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

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

```
## [1] 0.125
```

df=data.frame(a=1:10, b=11:20)