Class 06: R Functions

Yuhan Zhang (PID: A13829264)

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*.

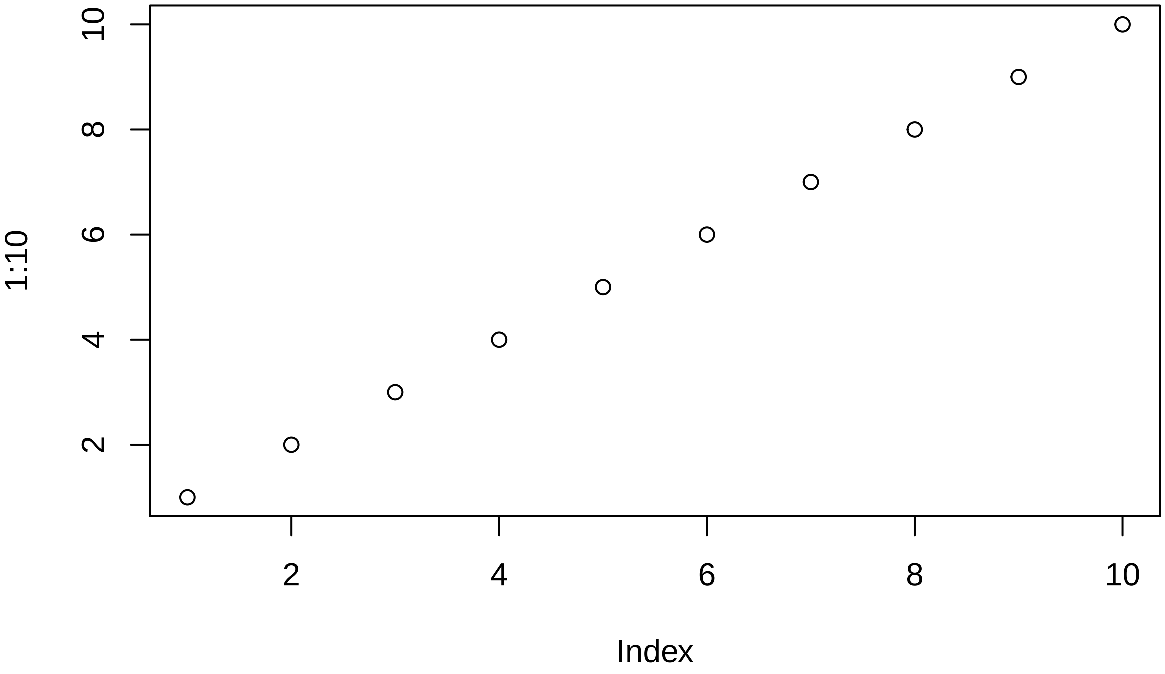
Do:

- this
- and that
- and another thing

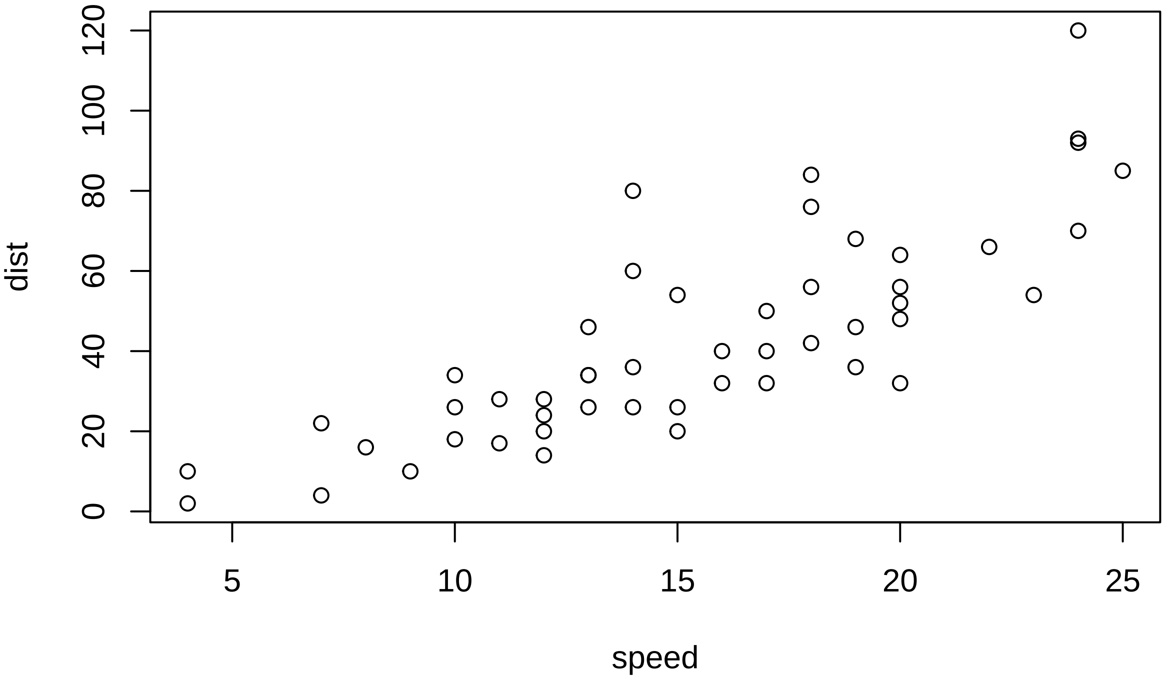
This is more text
and this is a new line

We can include some code:

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



```
# This is a comment and will not be passed to R
plot(cars)
```



Time to write the funciton

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]

```
# 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)
```

First I want to find the lowest score. I can use the **min()** to find it and the **which.min()** funciton to find where it si (i.e. its position in the vector)

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

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

I can use minus to get everything in the vector but hte lowest score.

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

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

Now i can call the **mean()** function to get the average.

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

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

Does this work for student2?

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

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

NO! Why not?

```r
student2
```

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

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

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

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

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

```r
student3
```

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

One great idea is to replace the NA values with zero. Let's do it.

The internet gods told me to try this

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

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

The is.na() function returns a logical vector where TRUE elements indicate the presence of NA values.

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

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

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

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

Let's replace NAs with zero

```r
student.prime <- student2
student.prime[is.na(student2)] = 0
student.prime
```

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

```r
x <- 1:5
x
```

```
## [1] 1 2 3 4 5
```

```r
x[2] = 100
x
```

```
## [1]   1 100   3   4   5
```

```r
x[3] = 200
x
```

```
## [1]   1 100 200   4   5
```

Ok we are so so close let's put these bits together to get our mean excluding the lowest score.

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

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

How about student3?

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

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

Great! We got it. this works. Let's simplify and make as clear as we can.

We can make the object names more clear

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

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

Doh! eeijt Barry entered the data 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
```

```
grade(student2)
```

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

```
grade(student3)
```

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

Now grade a whole class

First we got to read the gradebook for the class

```
url <- "https://tinyurl.com/gradeinput"
scores <- read.csv(url, 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
```

We are going to use the supper useful **apply()** funciton to grade all the students with our **grade()** function.

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

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

```
ans[which.max(ans)]
```

```
## student-18
##       94.5
```

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

Here I will use the **apply()** function again but this time look at the columns, which represent different homeworks.

```
apply(scores, 2, mean)
```

```
##  hw1  hw2  hw3  hw4  hw5
## 89.0   NA 80.8   NA   NA
```

I can ignore the NA missing values with na.rm = TRUE

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

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

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, na.rm = TRUE)
```

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

> **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
```

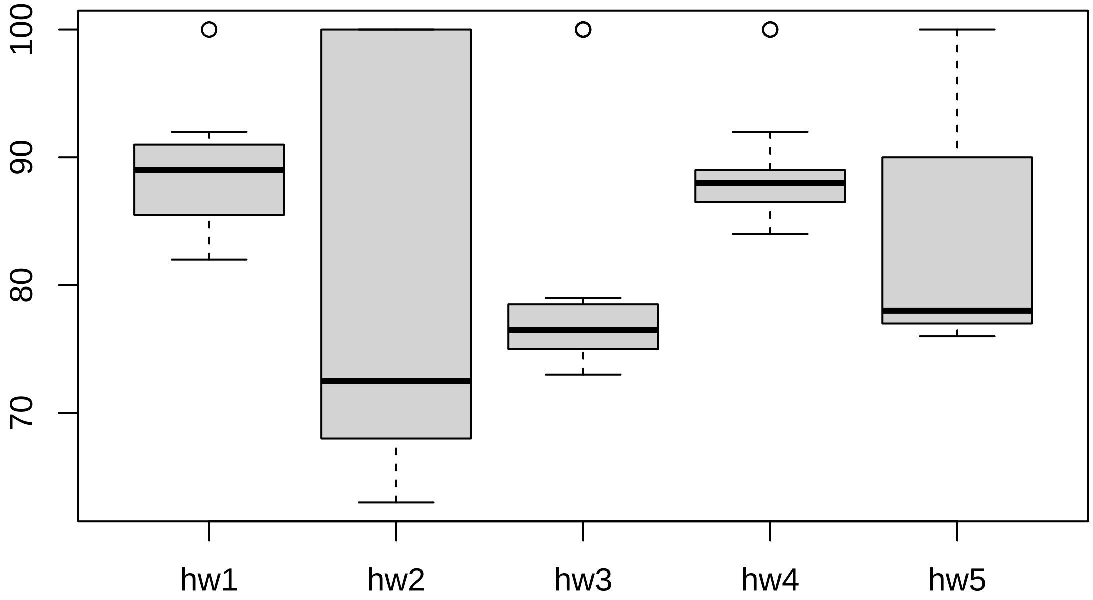
I can call the **cor()** for every homework and get a value for each but that sucks. Let's use **apply()** and do them all in one go.

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

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

Make a boxplot

```
boxplot(scores)
```


hw1
hw2
hw3
hw4
hw5