Class06:R_Functions

Zaida Rodriguez (PID:A59010549)

10/15/2021

Quick Rmarkdown intro

You can write text like any file. You can file text to be **bold** or *italic*.

Do:

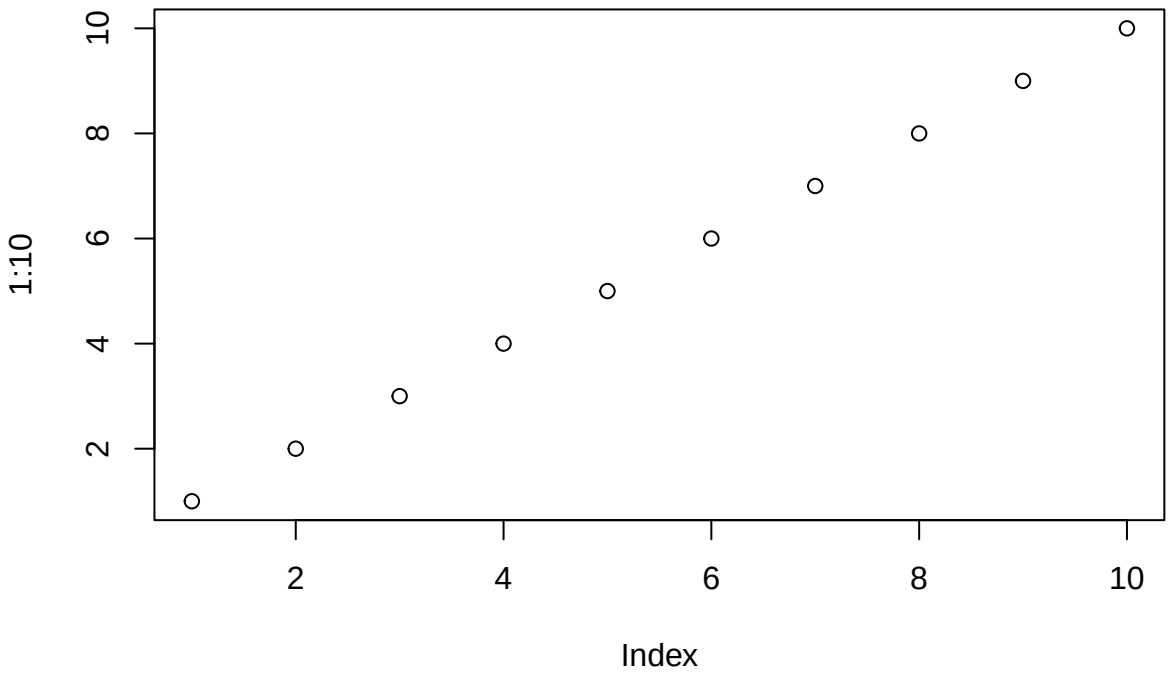
- this
- and that
- and another thing

This is more text

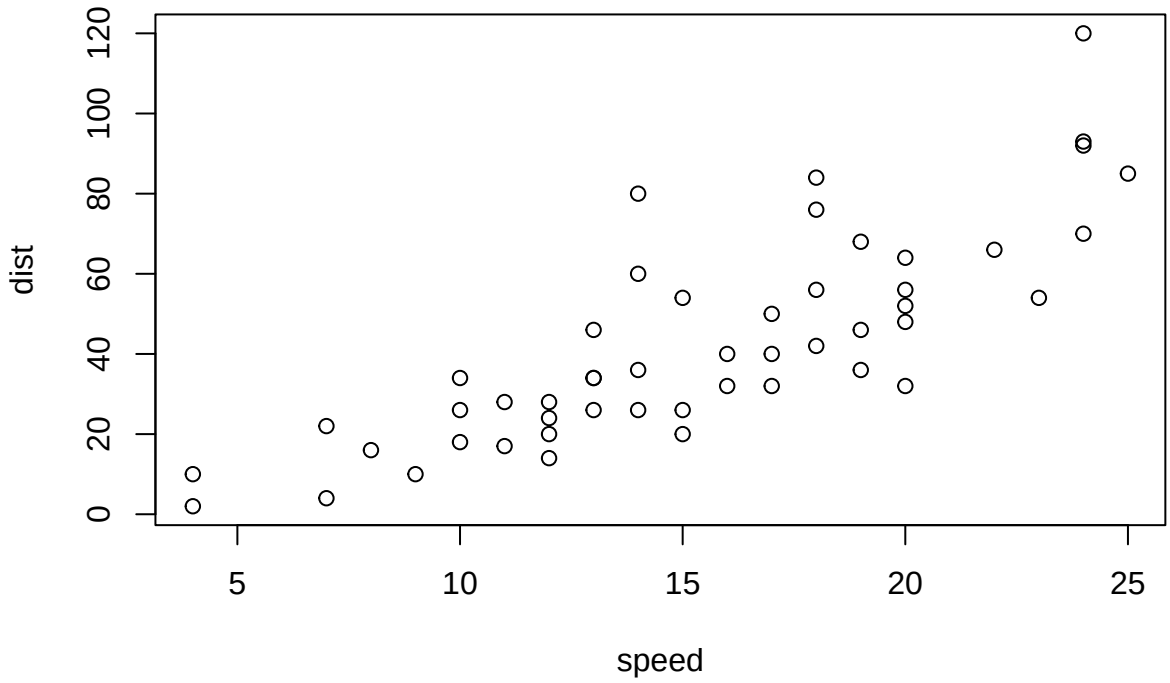
but in order to make a new line add two spaces

We can include some code:

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




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



Write a function

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 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 find the lowest score. You can use the **min()** to find it and the **which.min()** finds the position/location (position in vector).

```
min(student1)
```

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

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

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

You can omit the lowest score by **[-which.min(studentx)]. Overall, you can use minus to get everything in the vector but the 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
```

Now try it for student 2

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

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

This doesn't work on student 2. Dissect code and see where it is not working

```
student2
```

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

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

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

```
mean(student2)
```

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

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

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

If you use **na.rm = TRUE**, the system will omit NA's but the problem is that it will mess with your mean value, something you dont want.

Instead try changing NA to a zero value. If you don't know how to do this try googling it.

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

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

```
# this is telling us the position of the NA
```

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

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

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

```
#btw, if you put an ! in front of that function, it will flip you answers. In other words, the trues wi
```

Lets replace the NAs with zero by first copying the student2 data and overwriting in that value set in order to not destroy the original.

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

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

Now we can find the mean while excluding the lowest score

```
mean(student.prime[-which.min(student.prime)])
```

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

Now lets try on student 3

```
student.delta <- student3
student.delta[is.na(student.delta)] =0
student.delta
```

```
## [1] 90  0  0  0  0  0  0  0
```

```
mean(student.delta[-which.min(student.delta)])
```

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

Time to simplify the code snippet

We can make the object names clearer

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

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

Wait, the data 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 we can write out function. Make sure to input all 3 parts of a function

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

You could also highlight the code snippet you want, go to the **code** tab and then click on **extract function** and R will create the function after you name it.

After running the function, test to make sure it works

```
grade(student1)
```

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

```
grade(student2)
```

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

```
grade(student3)
```

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

##Now grade a whole class

First, read the gradebook for the class

```
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
```

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

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

The apply function grabs the entire matrix and by telling it 1, the function will be applied to all rows. If you wanted the columns you would tell it 2.

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

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

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

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

To find the homework, look at the columns using the **apply()** function.

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

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

An NA appears, you have to remove it by overwriting it with zero

```
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 try finding the lowest homework again.

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

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

The mean allows you to find the average in all columns to determine which one had the lowest scores

> Q4: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)?

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

```
cor(mask$hw1, ans)
```

```
## [1] 0.4250204
```

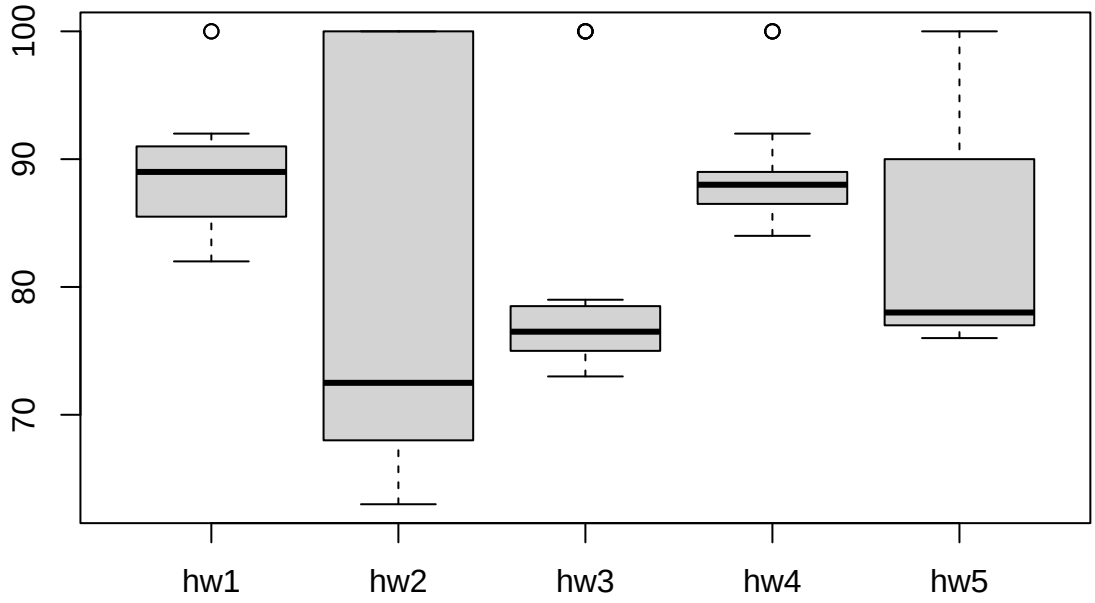
While you can use the **cor()** to each, the **apply()** will apply it to all the columns making your life so much easier

```
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)
```


100
90
80
70
hw1
hw2
hw3
hw4
hw5