

# Class 6: R Functions

Joanne Wu (PID: A17060517)

## Table of contents

A first silly function . . . . .	1
2nd example grade() function . . . . .	2

Functions are how we get work done in R. We call functions to do everything from reading data to doing analysis and outputting plots and results.

All functions in R have at least 3 things:

- a **name** (you get to pick this)
- input **arguments** (there can be only one or loads - again your call)
- the **body** (where the work gets done, this code between the curly brackets)

## A first silly function

Let's write a function to add some numbers. We can call it `add()`

```
x <- 10  
y <- 10  
x + y
```

```
[1] 20
```

```
add <- function(x) {  
  y <- 10  
  x + y  
}
```

Can I just use my new function?

```
add(1)
```

```
[1] 11
```

Let's make it a bit more flexible.

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

```
add(x=10, y=10)
```

```
[1] 20
```

```
add(10, 100)
```

```
[1] 110
```

## 2nd example grade() function

Write a functions to grade student work.

We will start with a simple version of the problem and the following example student vectors:

```
# 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)
```

Start with student1

```
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

```
mean(student3, na.rm=TRUE)
```

```
[1] 90
```

Ok lets try to work with student1 and find (and drop) the lowest score.

```
student1
```

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

Google told me about min() and max()

```
min(student1)
```

```
[1] 90
```

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

```
[1] 8
```

```
student1[8]
```

```
[1] 90
```

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

```
[1] 90
```

```
student1[-8]
```

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

Our first working snippet that drops the lowest score and calculates the mean

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

```
[1] 100
```

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

```
[1] 100
```

```
x <- student3  
mean(x[ -which.min(x) ], na.rm=T)
```

```
[1] NaN
```

Our approach to the NA problem (missing homeworks): We can replace all NA values with zero.

1st task is find the NA values (i.e. where are they in the vector)

```
x <- student2  
x
```

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

```
x == 90
```

```
[1] FALSE NA TRUE TRUE TRUE TRUE FALSE FALSE
```

```
x
```

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

```
is.na(x)
```

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

I have found the NA (TRUE) values from `is.na()` now I want to make them equal to zero (overwrite them/mask them etc.)

```
y <- 1:5  
y
```

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

```
y[y > 3] <- 0  
y
```

```
[1] 1 2 3 0 0
```

I want to combine the `is.na(x)` with making these elements equal to zero. And then take this “masked” (vector of student scores with NA values as zero) and drop the lowest and get the mean.

```
x <- student2  
x[is.na(x)] <- 0  
x
```

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

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

```
[1] 91
```

Now I can turn my most awesome snippet into my first function

```
grade <- function(x) {  
  # Make NA (missing work) equal to zero  
  x[is.na(x)] <- 0  
  # Drop lowest score and get mean  
  mean(x[ -which.min(x) ])  
}
```

```
grade(student2)
```

```
[1] 91
```

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 adequately 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]

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

	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

The `apply()` function in R is super useful but can be a little confusing to begin with. Lets have a look how it works.

```
ans <- apply(gradebook, 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
```

```
max(ans)
```

```
[1] 94.5
```

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

```
which.min( apply(gradebook, 2, mean, na.rm=TRUE) )
```

```
hw3  
3
```

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]

```
#ans  
cor(gradebook$hw1, ans)
```

```
[1] 0.4250204
```

```
#ans  
cor(gradebook$hw5, ans)
```

```
[1] NA
```

```
gradebook$hw5
```

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

Make all NA values into zero.

```
mask <- gradebook  
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(mask$hw5, ans)
```

```
[1] 0.6325982
```

Now we can use `apply()` to examine the correlation of every assignment in the masked gradebook to the overall score for each student in the class.

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

	hw1	hw2	hw3	hw4	hw5
	0.4250204	0.1767780	0.3042561	0.3810884	0.6325982

Q5. Make sure you save your Quarto document and can click the “Render” (or Rmarkdown”Knit”) button to generate a PDF format report without errors. Finally, submit your PDF to gradescope. [1pt]