# Class06

Troy Lee A17078296

Today we are going to explore R functions and begin to think about writting our own functions.

Let's start simple and write our first function to add some numbers.

Every function in R has at least 3 things:

- a **name**, we pick this
- one or more input **arguments**
- the **body**, where the work gets done

```
add <- function(x, y=1, z=0) {
  x + y
}
```

Now lets try it out

```
add(x=c(10,1,1,10),y=1)
```

```
[1] 11  2  2 11
```

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

```
[1] 20
```

```
add(10)
```

```
[1] 11
```

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

```
[1] 20
```

```
mean(c(10,10,NA), na.rm=T)
```

```
[1] 10
```

##Lab Sheet Work

> 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)
```

Begin by calculating the average for student1

```
student1
```

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

```
mean(student1)
```

```
[1] 98.75
```

Try on student2

```
student2
```

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

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

```
[1] 91
```

and student3

```
student3
```

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

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

```
[1] 90
```

Hmm…this sucks! I need to try something else and come back to this issue of missing values (NA)

We also want to drop the lowest score from a given students set of scores.

```
student1
```

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

```
student1[-8]
```

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

We can try the `min()` function to find the lowest score

```
min(student1)
```

```
[1] 90
```

I want to find the location of the min value not the value itself. For this I can use `which.min()`

```
student1
```

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

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

```
[1] 8
```

Let's put these two things together

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

```
[1] 100
```

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

```
[1] 100
```

We need to deal with NA (missing values) somehow?....

One idea is we make all the NA values zero.

```
x <- student2
x
```

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

```
x[2] <- 0
x
```

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

```
x <- student2
x
```

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

```
x[ is.na(x)] =0
x
```

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

So far we have a working snippet

```
x <- student1
##Find NAs in 'x' and make them 0
x[ is.na(x)] <- 0

# finds the min value and rm's it before getting mean
mean(x[-which.min(x)])
```

```
[1] 100
```

Now turn it into a function

```
grade <- function(x) {
  ##Find NAs in 'x' and make them 0
  x[ is.na(x)] <- 0

  # Drop lowest value and find mean
  mean(x[-which.min(x)])
}
```

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Now `apply()`to our gradebook

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

To use the `apply()` function on this `gradebook` dataset I need to decide whether I want to "apply" the `grade()` function over the rows or columns of the `gradebook`.

```
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
```

```
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]

```
apply(gradebook, 2, grade)
```

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

```
apply(gradebook, 2, mean, na.rm=T)
```

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

```
masked_gradebook <- gradebook
masked_gradebook [ is.na(masked_gradebook) ]= 0
apply(masked_gradebook, 2, mean)
```

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

I could modify the `grade()` function to do this too - i.e. not drop the lowest option

```
grade2 <- function(x, drop.low=TRUE) {
   ##Find NAs in 'x' and make them 0
  x[ is.na(x)] <- 0

  if(drop.low) {
    cat("Hello low")
    out <- mean(x[-which.min(x)])
} else {
    out <- mean(x)
    cat("No low")
  }
  return(out)

  # Drop lowest value and find mean
  mean(x[-which.min(x)])
}
```

```
grade2(student1, TRUE)
```

```
Hello low
```

```
[1] 100
```

> 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]

The function to calculate correlations in R is called `cor()`

```
x <- c(100,90,80,100)
y <- c(100,90,80,100)
z <- c(80,90,100,10)

cor(x,y,)
```

```
[1] 1
```

```
cor(x,z)
```

```
[1] -0.6822423
```

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

```
[1] 0.4250204
```

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

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

I want `apply()` the `cor()` function over the `masked_gradebook` and use the `ans` scores for the class