Unofficial Kahn Academy R Supplement: Mean, Median, and Mode

The KH lectures on centrality (mean, median, and mode) include a lot of basic arithmetic. It's worthwhile at this juncture to point out that, like nearly every other programming language, R can be used as a basic calculator. Furthermore, like all other object-oriented programming languages, R can store results for later use as objects.

```
sum_of_numbers<- 3 + 5+ 8 + 9
average_of_numbers= sum_of_numbers/4
sum_of_numbers
```

```
## [1] 25
```

```
average_of_numbers
```

```
## [1] 6.25
```

```
radius=2
sphere_surface_area<-4*pi*radius^2

print(paste("the average of this sequence of numbers is ", average_of_numbers))
```

```
## [1] "the average of this sequence of numbers is  6.25"
```

```
print(paste("the surface of this sphere is ", sphere_surface_area))
```

```
## [1] "the surface of this sphere is  50.2654824574367"
```

AYCS, here we calculated and stored a sum of numbers, then we found the average from that object. Finally, to show how R can use built-in constants, we calculated the surface of a sphere given a radius of 2. (Remember, when assigning objects you can either use "=" or "<-".) Finally, you can use the "print" command to display text with your results. This way, like a spreadsheet, you can change one number and if you re-run the command it will adjust all of the results accordingly.

Mean and median are incredibly simple to calculate in R, since their names are the exact names of the command (*mean* and *median*). Furthermore, the commands to find the mean and median are generally very similar across different programming languages. However, surprisingly there is no built-in command in R to find the mode. the *mode()* command returns the type of variable you have (with letters ("string" variable), with digits (a "numeric" variable), etc.). However, various R users have written functions that do this. A *function* is a command that allows R-users to write a user-written mini-command on the fly if they want to do something that R does not have a built-in command for. We will write our own *function* (or command) below. The details are not important at this point, suffice it to say that as you progress in your statistics education you will be forced to use user-defined functions more and more as the things you want to do with R become more complicated. Here we will draw on our miles per gallon example again to calculate the mean, median, and mode of our gas mileage. Once again we will have to create our new command, which we will label *Mode*. Because R is case sensitive (it matters whether you capitalize something or not), our *Mode* command is different from the built-in *mode* command.

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```
median(mtcars$mpg)
```

```
## [1] 19.2
```

```
Mode <- function(x) {
     ux <- unique(x)
     ux[which.max(tabulate(match(x, ux)))]
}
Mode(mtcars$mpg)
```

```
## [1] 21
```

AYCS, the mean, median, and mode for **mpg** is 20.1, 19.2, and 21, respectively. As a reminder, we can easily assign these values to objects to store them to make it more clear.

```
mean_mpg<-mean(mtcars$mpg)
median_mpg<-median(mtcars$mpg)
Mode <- function(x) {
     ux <- unique(x)
     ux[which.max(tabulate(match(x, ux)))]
}
mode_mpg<-Mode(mtcars$mpg)
```

Now all we have to do is type **mean_mph** and we should get the mean value for **mpg**.

Warning: if there are missing values, any of these commands will return a missing value (in R, missing values are represented by NA, in most other programs they are simply blank). So if, for example, we simply didn't know the miles per gallon for a Fiat 128 (in other words, if its value in the dataset was NA instead of 32.4), then simply running the mean command would return a response of "NA". To simply remove the missing values from consideration, we type the same command, but add *na.rm=TRUE*: *mean(mtcars$mpg, na.rm=TRUE)*. (I assume na.rm stands for "not applicable-remove," but I am not actually sure.) Conversely *na.rm=FALSE* means that we do not remove missing values. This is what R does automatically; however, so there is no reason to type it in.

As noted in KH, outliers can radically change the measures of centrality. While they are beyond the scope of what we are doing here, there are various math based tests that systematically detect data points that can be considered outliers. If you want to look more into this later on, Google "outlier detection in R" to get a sense of the different commands that have been written.