SIT718 Real World Analytics
Week 2 Practical

Existing functions

The mean and the median are already pre-programmed into R. We can calculate the arithmetic mean of a set of numbers using `mean()` where the argument is a vector. For example,

```
mean(c(2,3,6,7))
mean(2:7)
mean(my.seq)
mean(c(4,a,1:3,the.value,long.array))
```

Notice that in this last example, we could combine numbers, pre-assigned values and vectors to use as the input. We just need to remember to use `c()` and separate the values by a comma. The `mean()` function has some other optional parameters but we will not use these. If we entered something like `mean(3,2,51,2)`, the value returned would only be based on the first number, i.e. `3`, so we need to be careful.

The median function is also pre-programmed into R.

```
my.seq=array(c(1,2,3),200)

median(1:6)
median(c(1,7,82,2))
median(my.seq)
```

Some other pre-programmed functions that will be useful for us are `sum()`, `prod()`and `length()` which return the sum of the entries of the vector, the product of the values in the vector, and the length of the vector respectively.
We also have`min()`and`max()` functions which can take multiple arguments including a combination of single values or vectors and return the minimum and maximum arguments.

These can also be combined with the previous operations we looked at, so `sum(a^2)` would first square every value in a, and then add all these values together.

R Exercise 5 For the following vectors,

```
a <- c(1,8,3,9)
b <- c(2,2,1,1)
d <- c(3,4,6,81,9)
```

Evaluate:

Input	Expected output
`sum(a)`	`21`
`prod(d)`	`52488`
`length(b)`	`4`
`sum(a*b)`	`30`
`sum(a^b)`	`77`
`prod(a)^(1/length(b))`	`3.833659`
`min(d)`	`3`
`max(a,d)`	`81`
`min(max(a),max(b))`	`2`

Side Note 1.3 *Make sure you are careful with your brackets! The commands follow order of operations, which means without brackets in something like* `prod(a)^(1/length(b))`*, i.e. if you just wrote* `prod(a)^1/length(b)` *then we would have the product to the power of 1 (which is just the product), which is then divided by its length, so it would be* $(1 \times 8 \times 3 \times 9)/4 = 54$.

Creating new functions

In many situations it is convenient for us to be able to program our own functions. This is actually relatively easy to do in R and can become a very powerful tool!

Creation of a basic function essentially consists of 3 components:

- Pre-defining the function inputs
- A sequence of calculations
- Return of an output

SIT718 Real World Analytics
Week 2 Practical

We can start with an example that calculates the arithmetic mean (even though we don't need it because it is already defined in R with `mean()`).

```
our.mean <- function(x) {
sum(x)/length(x)
}
```

The value returned (i.e. the output) is always the calculation on the last line of our function before the `}` bracket. In this case, our whole calculation fits on one line, so we don't need to do any temporary assignment of values throughout. However, we could if we wanted to. Here is another example that will return the same output.

```
our.mean.2 <- function(x) {
n <- length(x)
s <- sum(x)
output <- s/n
output
}
```

In both of these examples, the `x` input is expected to be a vector. We could now calculate the mean using these R functions, e.g. by entering `our.mean(a)` or `our.mean(c(38,27,1))`. You will notice that the variables n and s are not saved in the R workspace (i.e. if you type 'n' and press enter, R will say that 'n' is not found.

Side Note 1.4 *Usually in programming we might be concerned about the efficiency of our steps. We won't focus on this at the moment, although it definitely can be important when we are dealing with large datasets.*

If we do an assignment on the last line of the function, no output will be returned, e.g.

```
nothing.happens <- function(x) {
sum(x) +2
a <- prod(x)
}
```

R will still make these calculations when you use the function, but it won't return any values.

Side Note 1.5 *So far, we have only considered one input into the function (although it is a multi-argument vector). In future topics we may need some other parameters.*

Let's now program functions for our geometric and harmonic means.

```
GM <- function(x) {
prod(x)^(1/length(x))
}

HM <- function(x) {
length(x)/sum(1/x)
}
```

Side Note 1.6 *When functions have just one line of calculation, there is no need for us to separate them like this. We use this format for clarity, however if we wanted to, we could enter the harmonic mean using*

```
HM <- function(x) { length(x)/sum(1/x)}
```

and the function would be the same. In some future functions, however, we will need several lines of calculation. When entering multi-line functions in R, you can either do this directly in the com- mand console (a '+' symbol will appear to let you know that you haven't closed off the necessary brackets), or you can hold the function in a library file and then copy and paste the whole sequence to the command console, or alternatively in support applications like ***R Studio*** *there are other ways to execute the code.*