Functions

Introduction to R for Public Health Researchers

Writing your own functions

This is a brief introduction. The syntax is:

```
functionName = function(inputs) {
< function body >
return(value)
}
```

Then you would run the 4 lines of the code, which adds it to your workspace.

Writing your own functions

Here we will write a function that returns the second element of a vector:

```
> return2 = function(x) {
+   return(x[2])
+ }
> return2(c(1,4,5,76))
```

```
[1] 4
```

Writing your own functions

Note that your function will automatically return the last line of code run:

```
> return2a = function(x) {
+   x[2]
+ }
> return2a(c(1,4,5,76))
```

```
[1] 4
```

And if your function is really one line or evaluation, like here, you do not need the curly brackets, and you can put everything on one line:

```
> return2b = function(x) x[2]
> return2b(c(1,4,5,76))
```

```
[1] 4
```

Writing your own functions

Also note that functions can take multiple inputs. Maybe you want users to select which element to extract

```
> return2c = function(x,n) x[n]
> return2c(c(1,4,5,76), 3)
```

```
[1] 5
```

Writing a simple function

Let's write a function, `sqdif`, that:

1. takes two numbers `x` and `y` with default values of 2 and 3.
2. takes the difference
3. squares this difference
4. then returns the final value

Writing a simple function

```
> sqdif <- function(x=2,y=3){
+       (x-y)^2
+ }
>
> sqdif()
```

```
[1] 1
```

```
> sqdif(x=10,y=5)
```

```
[1] 25
```

```
> sqdif(10,5)
```

```
[1] 25
```

Writing your own functions

Try to write a function called `top()` that takes a `matrix` or `data.frame`, and returns the first `n` rows and columns, with the default value of `n=5`.

Writing your own functions

Try to write a function called `top()` that takes a `matrix` or `data.frame`, and returns the first `n` rows and columns

```
> top = function(mat,n=5) mat[1:n,1:n]
> my.mat = matrix(1:1000,nr=100)
> top(my.mat) #note that we are using the default value fo
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1  101  201  301  401
[2,]    2  102  202  302  402
[3,]    3  103  203  303  403
[4,]    4  104  204  304  404
[5,]    5  105  205  305  405
```

Custom functions in apply

You can also designate functions "on the fly"

```
> matList = list(x = matrix(1:25,nc=5),y=matrix(26:50,nc=5)
> lapply(matList, function(x) x[1:2,1:2])
```

```
$x
     [,1] [,2]
[1,]    1    6
[2,]    2    7

$y
     [,1] [,2]
[1,]   26   31
[2,]   27   32
```

Simple apply

`sapply()` is a user-friendly version and wrapper of `lapply` by default returning a vector, matrix, or array

```
> sapply(matList, dim)
```

```
     x y
[1,] 5 5
[2,] 5 5
```

```
> sapply(matList, class)
```

```
        x         y
"matrix" "matrix"
```

```
> myList = list(a=1:10, b=c(2,4,5), c = c("a","b","c"),
+                     d = factor(c("boy","girl","girl")))
> tmp = lapply(myList,function(x) x[1])
> tmp

$a
[1] 1

$b
[1] 2

$c
[1] "a"

$d
[1] boy
Levels: boy girl

> sapply(tmp, class)
```

sapply can also be applied to columns of data frames

```
> library(readr)
> circ = read_csv(paste0("http://www.aejaffe.com/winterR_20
+   "data/Charm_City_Circulator_Ridership.csv"))
> sapply(circ,class)
```

```
              day              date  orangeBoardings orangeAl
     "character"      "character"         "integer"         "
   orangeAverage  purpleBoardings purpleAlightings     purpl
       "numeric"         "integer"         "integer"         "
  greenBoardings  greenAlightings     greenAverage  bannerB
       "integer"         "integer"         "numeric"         "
bannerAlightings    bannerAverage             daily
       "integer"         "numeric"         "numeric"
```