Goodbye loops, hello **`apply()`**

Nathaniel MacNell

EPID 799B

Fall 2016

Why **`apply()`**?

- You've been learning about R's ***abstraction***
  - The code is closer to how *you* might think
  - The computer does more "interpretation" of code
- Examples you've seen:
  - Objects [e.g. data frames, models, tables]
  - Functions handling multiple object types [e.g. plot()]
  - **`plyr`** data management
  - **`ggplot`** aesthetics
- **`apply()`** is R's abstraction for loops
  - Also a helpful "glue" to connect different functions

What is **`apply()`**, anyway?

- A family of functions used to perform repeated operations on objects.
  - Replaces loops, code also executes faster.
- Common apply() syntax elements:

`X`	The object you are doing something to
`MARGIN`	How to approach/scan the object
`FUN`	What to do in each step

Some apply() family members

Name	Input object	Output
apply()	**Matrix**	**Matrix or Vector**
sapply()	**List**	**Vector**
lapply()	List	List
vapply()	Vector	User defined
tapply()	*2 Vectors*	*List*
mapply()	*Multiple objects*	*Matrix or Vector*
rapply()	*Nested list*	*Nested list*

Most useful Sometimes useful *Rarely used*

Starting off with **sapply(X, FUN)**

- sapply() can be used to replace simple for() loops:

```
x <- 1:100
for(i in 1:length(x) ) {
  x <- x^2
}
```

```
x <- 1:100
square <- function(x) {
   x^2
}
sapply(x, square)
```

```
x <- 1:100
sapply(x, function(i) i^2)
```

Creating a practice apply() matrix

```
x <- matrix(1:100,nrow=10)
> x
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100
>
```

apply(X, MARGIN, FUN)

- Three options, based on setting the margin:

MARGIN=1	By row
MARGIN=2	By Column
MARGIN=c(1,2)	By Cell

```
> apply(x,1,sum)
 [1] 460 470 480 490 500 510 520 530 540 550
> apply(x,2,sum)
 [1]  55 155 255 355 455 555 655 755 855 955
```

The apply() extended family

Name	What it does
aggregate()	Applies a function to a vector by a group (i.e. find the mean age by treatment group).
by()	Apply function to a data frame by a grouping factor (i.e. perform analysis on subsets of dataset).
Map()	Apply functions **across** multiple variables (i.e. average blood pressure values).
replicate()	Repeats the same function many times (good for simulations).

Practice!

Variables 83 through 88 in the births dataset are indicators of maternal morbidity.

1. Create a data frame of only these variables.
2. Make one table of these variables using apply()
3. Find the number of Y values in each row using apply(). Make this a variable called morbid.
4. Use aggregate to find the average number of morbidities by a maternal characteristic of your choice (for example, TOTPREG).