



# Introduction to the R Language

## Loop Functions

Roger Peng, Associate Professor  
Johns Hopkins Bloomberg School of Public Health

# Looping on the Command Line

Writing `for`, `while` loops is useful when programming but not particularly easy when working interactively on the command line. There are some functions which implement looping to make life easier.

- `lapply`: Loop over a list and evaluate a function on each element
- `sapply`: Same as `lapply` but try to simplify the result
- `apply`: Apply a function over the margins of an array
- `tapply`: Apply a function over subsets of a vector
- `mapply`: Multivariate version of `lapply`

An auxiliary function `split` is also useful, particularly in conjunction with `lapply`.

# lapply

`lapply` takes three arguments: a list `x`, a function (or the name of a function) `FUN`, and other arguments via its `...` argument. If `x` is not a list, it will be coerced to a list using `as.list`.

```
> lapply
function (X, FUN, ...)
{
  FUN <- match.fun(FUN)
  if (!is.vector(X) || is.object(X))
    X <- as.list(X)
  .Internal(lapply(X, FUN))
}
```

The actual looping is done internally in C code.

# lapply

`lapply` always returns a list, regardless of the class of the input.

```
> x <- list(a = 1:5, b = rnorm(10))  
> lapply(x, mean)  
$a  
[1] 3  
  
$b  
[1] 0.0296824
```

# lapply

```
> x <- list(a = 1:4, b = rnorm(10), c = rnorm(20, 1), d = rnorm(100, 5))
> lapply(x, mean)
$a
[1] 2.5

$b
[1] 0.06082667

$c
[1] 1.467083

$d
[1] 5.074749
```

# lapply

```
> x <- 1:4
> lapply(x, runif)
[[1]]
[1] 0.2675082

[[2]]
[1] 0.2186453 0.5167968

[[3]]
[1] 0.2689506 0.1811683 0.5185761

[[4]]
[1] 0.5627829 0.1291569 0.2563676 0.7179353
```

# lapply

```
> x <- 1:4
> lapply(x, runif, min = 0, max = 10)
[[1]]
[1] 3.302142

[[2]]
[1] 6.848960 7.195282

[[3]]
[1] 3.5031416 0.8465707 9.7421014

[[4]]
[1] 1.195114 3.594027 2.930794 2.766946
```

# lapply

`lapply` and friends make heavy use of *anonymous* functions.

```
> x <- list(a = matrix(1:4, 2, 2), b = matrix(1:6, 3, 2))
> x
$a
      [,1] [,2]
[1,]    1    3
[2,]    2    4

$b
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```



# lapply

An anonymous function for extracting the first column of each matrix.

```
> lapply(x, function(elt) elt[,1])  
$a  
[1] 1 2  
  
$b  
[1] 1 2 3
```

# sapply

`sapply` will try to simplify the result of `lapply` if possible.

- If the result is a list where every element is length 1, then a vector is returned
- If the result is a list where every element is a vector of the same length ( $> 1$ ), a matrix is returned.
- If it can't figure things out, a list is returned

# sapply

```
> x <- list(a = 1:4, b = rnorm(10), c = rnorm(20, 1), d = rnorm(100, 5))
> lapply(x, mean)
$a
[1] 2.5

$b
[1] 0.06082667

$c
[1] 1.467083

$d
[1] 5.074749
```

# sapply

```
> sapply(x, mean)
      a      b      c      d
2.50000000 0.06082667 1.46708277 5.07474950

> mean(x)
[1] NA
Warning message:
In mean.default(x) : argument is not numeric or logical: returning NA
```