Week 3: Loop Functions and Debugging

Objectives

- Define an anonymous function and describe its use in loop functions
- Describe how to start the R debugger for an arbitrary R function
- Describe what the traceback() function does and what is the function(stack)

Loop Functions

lapply - Loop over a list and evaluate a function on each element split is helpful with lappy sapply - Same as lappy 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 (table apply) mapply - multivariate version of lappy

lapply

Takes three arguments 1. List x 2. function 3. other arguments Want to apply a function to every part of a list. ALWAYS RETURNS A LIST

Example 1: lapply

```
x <- list (a = 1:5, b = rnorm(10))
lapply(x, mean)

## $a
## [1] 3
##
## $b
## [1] 0.4369535
```

Example 2: 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] 0.09985579
##
## $c
## [1] 1.219963
##
## $d
## [1] 5.042146
```

Example 3: lapply

```
x <- 1:4
lapply(x, runif)

## [[1]]
## [1] 0.5394944
##
## [[2]]
## [1] 0.1757852 0.7786750
##
## [[3]]
## [1] 0.5159929 0.1767363 0.2092623
##
## [[4]]
## [1] 0.7808720 0.0869518 0.6781416 0.1715009

#runif generates random numbers of same vector length
```

Example 4: lapply matrices

```
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(x , function(elt) elt[,1])

## $a
## [1] 1 2
##
## $b
## [1] 1 2 3
```

Example 5: sapply

```
x <- list(a = 1:4, b = rnorm(10), c = rnorm(20,1), d = rnorm(100,5))
sapply(x,mean)

##          a          b          c          d
## 2.50000000 0.06327598 1.27846343 5.14536243
```

- If the result is a list where very 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.

Example 6: apply

```
x <- matrix(rnorm(200), 20,10)
apply(x, 2, mean)

##  [1]  0.180824193  0.044504731  0.008323868 -0.099903610 -0.272981271
##  [6] -0.333072619 -0.157402196  0.046872968  0.015320418 -0.109851882

apply(x, 1, sum)

##  [1] -1.699515463 -4.226841895  0.159136003 -2.902936955 -2.469584203
##  [6] -0.366359681 -1.621644121  0.034001443 -2.059856005  2.406997172
## [11]  0.353939936  2.025993800  3.663958443  2.696915539  0.002318475
## [16]  1.666218301 -0.458080374 -3.546147394 -3.452699951 -3.753121048
```

- It is most often apply to a function to the rows or columns of a matrix.
- It can be used with general arrays.
- No faster than writing a loop, but only takes one line.

Col/Row Sums and Means - rowSums = apply(x, 1, sum) rowsum(x, dim = 2) - rowMeans = apply(x, 1, mean) - colSums = apply(x, 2, sum) - colMeans = apply(x, 2, mean) MUCH FASTER THAN apply ^

Example 7: apply - quantiles

```
x <- matrix(rnorm(200), 20,10)
apply(x, 1, quantile, probs = c(0.25,0.75))

##           [,1]       [,2]      [,3]       [,4]       [,5]       [,6]
## 25% -0.8134744 -0.6544722 -0.827400 -0.9379143 -0.2975042 -1.1932991
## 75%  1.1151379  0.5406410  1.102991  0.9899774  0.7353574  0.1579064
##           [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
## 25% -0.3313599 -0.9730072 -0.8047646 -0.7934514 -1.2771751 -0.3610043
## 75%  0.5197522  0.1284218  1.1106553  0.1728659  0.4234457  0.9909073
##            [,13]     [,14]       [,15]      [,16]      [,17]      [,18]
## 25% -0.007159517 0.3025448 -1.11512784 -0.8185398 -0.2586587 -0.8054323
## 75%  0.629374658 0.8425538  0.06285146  0.1571480  0.8934436  0.7016651
##          [,19]      [,20]
## 25% -0.3877618 -0.2987055
## 75%  0.3893132  0.8079234
```

Example 8: mapply

```
mapply(rep, 1:4, 4:1)

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

```
## [[2]]
## [1] 2 2 2
##
## [[3]]
## [1] 3 3
##
## [[4]]
## [1] 4
```

Example 9: tapply - take group means

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10)
tapply(x,f,mean)
```

```
##          1          2          3
## 0.08361959 0.45907639 1.19020946
```

```
tapply(x,f,range)
```

```
## $`1`
## [1] -0.8690924  1.4477650
##
## $`2`
## [1] 0.06423904 0.90350077
##
## $`3`
## [1] -0.3975269  2.7178924
```

Example 10: split

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10)
split(x,f)
```

```
## $`1`
##  [1]  1.31048428  0.80749459 -0.51456559  0.46400430 -0.38137306
##  [6] -0.02299561 -0.31672866  0.38821793 -0.62414916 -0.15988740
##
## $`2`
##  [1] 0.35255447 0.34139672 0.88161912 0.03733652 0.47214916 0.34139770
##  [7] 0.04461047 0.15769601 0.01549005 0.09736094
##
## $`3`
##  [1] -1.74624279 -0.02235782  1.14947983  1.15988746  1.70931678
##  [6]  0.42531853  1.54256880  1.01757626  2.07261225  0.58197641
```

```
lapply(split(x,f),mean)
```

```
## $`1`
## [1] 0.09505016
##
```

```
## $`2`
## [1] 0.2741611
##
## $`3`
## [1] 0.7890136
```

Example 11: split a data frame/ split by month

```
library(datasets)
head(airquality)

##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6

s <- split(airquality, airquality$Month)
lapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R","Wind")]))

## $`5`
##    Ozone  Solar.R     Wind
##       NA       NA 11.62258
##
## $`6`
##     Ozone   Solar.R      Wind
##        NA 190.16667  10.26667
##
## $`7`
##      Ozone    Solar.R       Wind
##         NA 216.483871   8.941935
##
## $`8`
##    Ozone  Solar.R     Wind
##       NA       NA 8.793548
##
## $`9`
##    Ozone  Solar.R     Wind
##       NA 167.4333  10.1800

lapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R","Wind")], na.rm =
TRUE))

## $`5`
##     Ozone   Solar.R      Wind
##  23.61538 181.29630  11.62258
##
## $`6`
##     Ozone   Solar.R      Wind
##  29.44444 190.16667  10.26667
##
## $`7`
##      Ozone    Solar.R       Wind
##  59.115385 216.483871   8.941935
##
## $`8`
##      Ozone    Solar.R       Wind
##  59.961538 171.857143   8.793548
##
## $`9`
##     Ozone   Solar.R      Wind
##  31.44828 167.43333  10.18000
```

Debug

printmessage() traceback() debug() browser() trace() recover()