STAT 345 Lab Eight

Ian Roberts

## Bug Hunting

In this lab, you will fix a couple of buggy function definitions. Probably the easiest workflow is to define the function in your console, and then run the sample commands—they will either give errors or produce the wrong outputs. Using any combination of: reading the error messages, traceback(), and cat() or print(), you must find and fix the bugs. Sometimes it can also help to try multiple different inputs, i.e., try new function calls, rather than just looking at the sample calls given to you, in order to determine the bugs. You shouldn’t show any of your debugging work in your final knitted answers—so, don’t show calls to traceback(), and don’t leave any cat() or print() calls in the final, fixed function. (You don’t have to do anything yet, this was just to setup this section of the lab.)

Submit your knitted file to Canvas. There’s no “quiz” for this lab!

1. Below is a function called get.cols.with.ab.zeros(), but it has a few bugs. A few sample matrices are given below in mat, identity.mat, along with some sample calls that give errors. After fixing the bugs, the calls to get.cols.with.ab.zeros() should produce the outputs as described in comments.

# Function: cols.with.ab.zeros, to retrieve columns of matrix that have between  
# a and b zeros, each  
# Inputs:  
# - my.mat: the original matrix   
# - a: lower bound for number of zeros allowed; default is 0  
# - b: upper bound for number of zeros allowed; default is Inf  
# Output: the new matrix  
  
cols.with.ab.zeros <- function(my.mat, a=0, b=Inf) {  
 zeros.per.column <- colSums(my.mat == 0)  
 i.to.keep <- c()  
 for(i in 1:length(zeros.per.column)) {  
 i.to.keep <- c(i.to.keep,zeros.per.column[i] >= a && zeros.per.column[i] <= b)  
 }  
 return(my.mat[,i.to.keep, drop = FALSE])  
}  
  
mat = matrix(c(0,0,1,0,1,1,1,1,1), 3, 3)  
identity.mat = diag(1, 3)  
cols.with.ab.zeros(mat) # Should get back original matrix

## [,1] [,2] [,3]  
## [1,] 0 0 1  
## [2,] 0 1 1  
## [3,] 1 1 1

cols.with.ab.zeros(mat, a=1, b=2) # Should get back first 2 columns of mat

## [,1] [,2]  
## [1,] 0 0  
## [2,] 0 1  
## [3,] 1 1

cols.with.ab.zeros(mat, a=2, b=2) # Should get just 1st column of mat; note

## [,1]  
## [1,] 0  
## [2,] 0  
## [3,] 1

# this should still be a matrix though, and not a numeric vector!  
cols.with.ab.zeros(identity.mat, a=2, b=2) # Should get back original matrix

## [,1] [,2] [,3]  
## [1,] 1 0 0  
## [2,] 0 1 0  
## [3,] 0 0 1

1. Below is a function called list.extractor(), but it has a few bugs. A sample list is given below in cool.list, along with some sample calls that give errors. After fixing the bugs, the calls to list.extractor() should produce the outputs as described in comments. (As a partial hint, check out the yellow, triangle warning signs that RStudio adds to the line numbers!)

# Function: list.extractor, to extract elements of a list  
# Inputs:  
# - my.list: the original list   
# - i.to.keep: vector of indices, corresponding to elements of the list we  
# want to keep. Default is NULL, in which case this argument is ignored  
# - i.to.remove: vector of indices, corresponding to elements of the list we  
# want to remove Default is NULL, in which case this argument is ignored.  
# NOTE: if both i.to.keep and i.to.remove are non-NULL, then the first   
# one should take precedence (i.e., we don't remove anything)  
# Output: the new list  
  
list.extractor = function(my.list, i.to.keep=NULL, i.to.remove=NULL) {  
 if (!is.null(i.to.keep) && !is.null(i.to.remove)) {  
 L = my.list[i.to.keep]  
 } else if (!is.null(i.to.keep)) {  
 L = my.list[i.to.keep]  
 } else if (!is.null(i.to.remove)) {  
 L = my.list[-i.to.remove]  
 }  
 return(L)  
}  
  
cool.list = list(ints=1:10, lets=letters[1:8], fracs=1:7/7,  
 bools=sample(c(TRUE,FALSE), 5, replace=TRUE))  
list.extractor(cool.list, i.to.keep=c(1,3)) # Should get list with ints, fracs

## $ints  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $fracs  
## [1] 0.1428571 0.2857143 0.4285714 0.5714286 0.7142857 0.8571429 1.0000000

list.extractor(cool.list, i.to.remove=4) # Should get list without bools

## $ints  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $lets  
## [1] "a" "b" "c" "d" "e" "f" "g" "h"  
##   
## $fracs  
## [1] 0.1428571 0.2857143 0.4285714 0.5714286 0.7142857 0.8571429 1.0000000

list.extractor(cool.list, i.to.keep=2:4, i.to.remove=4) # Should get list with

## $lets  
## [1] "a" "b" "c" "d" "e" "f" "g" "h"  
##   
## $fracs  
## [1] 0.1428571 0.2857143 0.4285714 0.5714286 0.7142857 0.8571429 1.0000000  
##   
## $bools  
## [1] TRUE TRUE TRUE TRUE FALSE

# lets, fracs, and bools (the i.to.remove argument should be ignored)  
# I did it!