**MODULE # 11 DEBUGGING AND DEFENSIVE PROGRAMMING IN R**

The code below contains a 'deliberate' bug that must be rectified.

tukey\_multiple <- function(x) {  
  outliers <- array(TRUE,dim=dim(x))  
  for (j in 1:ncol(x))  
  {  
    outliers[,j] <- outliers[,j] && tukey.outlier(x[,j])  
  }  
  outlier.vec <- vector(length=nrow(x))  
  for (i in 1:nrow(x))  
  { outlier.vec[i] <- all(outliers[i,]) } return(outlier.vec) }

When the above code is run, it will result in the following error:

Error: unexpected symbol in:  
" for (i in 1:nrow(x))  
{ outlier.vec[i] <- all(outliers[i,]) } return"

The function 'traceback' prints the list of functions that were called before the error occurred.

The function 'source', which reads and executes the code, can be used to find the exact location of the error, which will help to rectify the error.

So, when the 'source' function is implemented, it gives the location of the error just before 'return'

> traceback("tukey\_multiple")

1: tukey\_multiple

> source("tukey\_multiple.R")

Error in source("tukey\_multiple.R") :

tukey\_multiple.R:9:43: unexpected symbol

8: for (i in 1:nrow(x))

9: { outlier.vec[i] <- all(outliers[i,]) } return

^

This error can be be overcome by placing 'return(outlier.vec)}' in the next line which will isolate

it from 'for()' loop.

tukey\_multiple <- function(x) {  
outliers <- array(TRUE,dim=dim(x))  
for (j in 1:ncol(x))  
{  
outliers[,j] <- outliers[,j] && tukey.outlier(x[,j])  
}  
outlier.vec <- vector(length=nrow(x))  
for (i in 1:nrow(x))  
{ outlier.vec[i] <- all(outliers[i,]) }   
return(outlier.vec) }

When 'debug' function is implemented on the code 'tukey\_multiple',following error is

noted which states that 'x' is missing. So, 'x' argument needs to be defined.

> debug(tukey\_multiple)

> tukey\_multiple()

debugging in: tukey\_multiple()

debug at #1: {

outliers <- array(TRUE, dim = dim(x))

for (j in 1:ncol(x)) {

outliers[, j] <- outliers[, j] && tukey.outlier(x[, j])

}

outlier.vec <- vector(length = nrow(x))

for (i in 1:nrow(x)) {

outlier.vec[i] <- all(outliers[i, ])

}

return(outlier.vec)

}

Browse[2]> n

debug at #2: outliers <- array(TRUE, dim = dim(x))

Browse[2]> n

Error in array(TRUE, dim = dim(x)) :

argument "x" is missing, with no default

Using function 'matrix' is implemented, a 3\*4 matrix is generated with normally random distributed values. Function 'debug' is implemented on the code followed by calling function 'tukey\_multiple' with matrix.

> matrix <- matrix(rnorm(12), nrow = 3, ncol = 4)

> debug(tukey\_multiple)

> tukey\_multiple(matrix)

debugging in: tukey\_multiple(matrix)

debug at #1: {

outliers <- array(TRUE, dim = dim(x))

for (j in 1:ncol(x)) {

outliers[, j] <- outliers[, j] && tukey.outlier(x[, j])

}

outlier.vec <- vector(length = nrow(x))

for (i in 1:nrow(x)) {

outlier.vec[i] <- all(outliers[i, ])

}

return(outlier.vec)

|  |
| --- |
| }  Browse[2]>  The result of debug() presents with 'Browse[2]>',  Typing n executes the current line and prints the next one  Typing c executes the result of the function without stopping  Typing Q quits the debugging  Typing where tells where you are in the function call stack  Typing ls() to list all the object in the local environment  Typing an object or print (<object name>) tells you current valve of  the object. |
| Browse[2]> n  debug at #2: outliers <- array(TRUE, dim = dim(x)) Browse[2]> n  debug at #3: for (j in 1:ncol(x)) {  outliers[, j] <- outliers[, j] && tukey.outlier(x[, j]) } Browse[2]> n  debug at #5: outliers[, j] <- outliers[, j] && tukey.outlier(x[, j]) Browse[2]> n  Error in tukey.outlier(x[, j]) : could not find function "tukey.outlier" In addition: Warning message: In outliers[, j] && tukey.outlier(x[, j]) :  'length(x) = 3 > 1' in coercion to 'logical(1)'  The above error mentions that 'tukey.outlier' cannot be found.  So, eliminating the function '&& tukey.outlier(x[,j])',  we implement function 'debug' again on the new code. |

> debug(tukey\_multiple)

> tukey\_multiple(matrix)

debugging in: tukey\_multiple(matrix)

debug at tukey\_multiple.R#1: {

outliers <- array(TRUE, dim = dim(x))

for (j in 1:ncol(x)) {

outliers[, j] <- outliers[, j]

}

outlier.vec <- vector(length = nrow(x))

for (i in 1:nrow(x)) {

outlier.vec[i] <- all(outliers[i, ])

}

return(outlier.vec)

}

Browse[2]> n

debug at tukey\_multiple.R#2: outliers <- array(TRUE, dim = dim(x))

Browse[2]> n

debug at tukey\_multiple.R#3: for (j in 1:ncol(x)) {

outliers[, j] <- outliers[, j]

}

Browse[2]> n

debug at tukey\_multiple.R#5: outliers[, j] <- outliers[, j]

Browse[2]> n

debug at tukey\_multiple.R#5: outliers[, j] <- outliers[, j]

Browse[2]> n

debug at tukey\_multiple.R#5: outliers[, j] <- outliers[, j]

Browse[2]> n

debug at tukey\_multiple.R#5: outliers[, j] <- outliers[, j]

Browse[2]> n

debug at tukey\_multiple.R#7: outlier.vec <- vector(length = nrow(x))

Browse[2]> n

debug at tukey\_multiple.R#8: for (i in 1:nrow(x)) {

outlier.vec[i] <- all(outliers[i, ])

}

Browse[2]> n

debug at tukey\_multiple.R#9: outlier.vec[i] <- all(outliers[i, ])

Browse[2]> n

debug at tukey\_multiple.R#9: outlier.vec[i] <- all(outliers[i, ])

Browse[2]> n

debug at tukey\_multiple.R#9: outlier.vec[i] <- all(outliers[i, ])

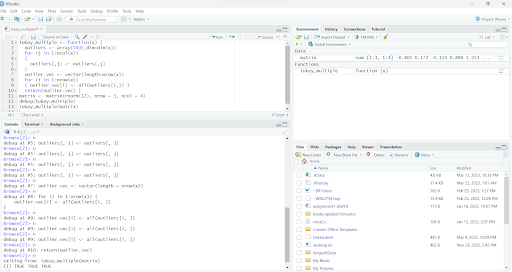
Browse[2]> n

debug at tukey\_multiple.R#10: return(outlier.vec)

Browse[2]> n

exiting from: tukey\_multiple(matrix)

[1] TRUE TRUE TRUE

[](https://blogger.googleusercontent.com/img/a/AVvXsEi3gWIMkuYMvgkNJsDOkpB9Ei3YQFvGP0fcwSAwDBsZgc_CINokLbXKGzztm2Rasu-F41dR0J__zoIl2oX_3YQqQSGQa5n_oLE3sePIqrTCshXgxux2zVBxjcnBwEnETC_5ekbcyeXvN0H96So0LZPsNYvTV5s55ZTyZDPH_J2bELL69IQE3OVATj3D)

In this way, implementing above codes and rectifying all the errors, a code can be DEBUGGED.

*URL to git repo:* [*https://github.com/VedaVangala/vedas-r-repo/tree/main/R11*](https://github.com/VedaVangala/vedas-r-repo/tree/main/R11)

**References**

Wickham, H. (2019). *Debugging, condition handling, and defensive programming. Advanced R.*CRC Press, Boca Raton, FL (2019). ISBN: 978-0815384571. [*http://adv-r.had.co.nz/Exceptions-Debugging.html*](https://www.blogger.com/blog/post/edit/2482057455959808426/2857962094230595299)

Matloff. N. (2011). *The Artof R programming*.(Chapter 13)