**Debugging and defensive programming in R**

**1. The code below contains a 'deliberate' bug!**

**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) }**

> Find the bug and fix it.

**SOLUTION:** This module centers around the process of debugging R code. Specifically, it focuses on the debugging of functions, which will be extremely important for package creation. Using the function given above, the goal is to debug and fix the code in order for the function to work.

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

When attempting to run the ‘tukey\_multiple()’ function, an error occurred during the compilation process. The error message indicated that there was a problem with the code on a specific line, and upon examining the code, it was discovered that the ‘return()’ statement was on the same line as the ‘for()’ loop. This may have confused R, leading to the error. By moving the ‘return()’ statement to the next line, the error was resolved and the function could be compiled successfully. It is important to ensure that code is structured correctly in order to avoid such errors.

To start debugging the tukey\_multiple() function, a dummy matrix was created to provide input with dimensions as the second line of the function requires an input with dim. It was also tested with a dummy array but it resulted in an "incorrect number of dimensions" error when used as input. If an input without dimensions is used, an error will be thrown at the second line of the function.

|  |
| --- |
| a <- matrix(c(1,2,3,4,5,6,7,8,9,10), nrow = 5, ncol = 2)  debug(tukey\_multiple)  tukey\_multiple(a) |

Once the dummy matrix was created and used as input for the tukey\_multiple() function, the code was examined line-by-line,

The absence of the defined function tukey.outlier() caused an error that prevented the smooth execution of tukey\_multiple(), which led to its omission in the next iteration for the purpose of ensuring the code runs properly.

|  |
| --- |
| tukey\_multiple <- function(x) {  outliers <- array(TRUE, dim = dim(x) + 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)  }  a <- matrix(c(1,2,3,4,5,6,7,8,9,10), nrow = 5, ncol = 2)  debug(tukey\_multiple)  tukey\_multiple(a) |

And the output is

|  |
| --- |
| tukey\_multiple <- function(x) {  + outliers <- array(TRUE, dim = dim(x) + 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)  + }  > a <- matrix(c(1,2,3,4,5,6,7,8,9,10), nrow = 5, ncol = 2)  > debug(tukey\_multiple)  > tukey\_multiple(a)  debugging in: tukey\_multiple(a)  debug at #1: {  outliers <- array(TRUE, dim = dim(x) + 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]> function(x) {  + outliers <- array(TRUE, dim = dim(x) + 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)  + }  function(x) {  outliers <- array(TRUE, dim = dim(x) + 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)  }  <environment: 0x0000013a14176d18>  Browse[2]>  debug at #2: outliers <- array(TRUE, dim = dim(x) + dim(x))  Browse[2]>  debug at #3: for (j in 1:ncol(x)) {  outliers[, j] <- outliers[, j]  }  Browse[2]>  debug at #4: outliers[, j] <- outliers[, j]  Browse[2]>  debug at #4: outliers[, j] <- outliers[, j]  Browse[2]>  debug at #6: outlier.vec <- vector(length = nrow(x))  Browse[2]>  debug at #7: for (i in 1:nrow(x)) {  outlier.vec[i] <- all(outliers[i, ])  }  Browse[2]>  debug at #8: outlier.vec[i] <- all(outliers[i, ])  Browse[2]>  debug at #8: outlier.vec[i] <- all(outliers[i, ])  Browse[2]>  debug at #8: outlier.vec[i] <- all(outliers[i, ])  Browse[2]>  debug at #8: outlier.vec[i] <- all(outliers[i, ])  Browse[2]>  debug at #8: outlier.vec[i] <- all(outliers[i, ])  Browse[2]>  debug at #10: return(outlier.vec)  Browse[2]>  exiting from: tukey\_multiple(a)  [1] TRUE TRUE TRUE TRUE TRUE  >  > |

As shown above, when debugging the revised function omitting tukey.outlier(), the tukey\_multiple() function is able to run. Overall, the debugging experience was definitely something new to me. As I have been working with R for quite some time now, syntax-related errors were easier to see. This module definitely reminded me to pay much closer attention to what is actually written in the code, as I had fallen for the simple trap of inputting a single number, completely ignoring the array/dimension portion in line 2. After that little misstep I was able to detect the undefined function and remove it. One thing to note is that I wish I understood the goal of the function better, as it would also make certain whether or not outright removal of the tukey.outlier() function was the right call, or whether or not more should have been added in order to define the function. In either case, the function seems to work and shows no errors in debug mode, so for now I am satisfied.