STAT 33B Lab Workbook Wk 11

Won Shil Park (3033452021)

Apr 4, 2021

This workbook is due **Apr 8, 2021** by 9:00am PT or by midnight Apr 9 if you are attending lab.

- Knit and submit the generated PDF file on Gradescope.

Fixing Buggy Code

Here is a function with many bugs. You will uncover them one at a time and correct the function in stages.

```
whisker.endpoints = function(x) {
  if (is.na(x) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = FALSE)
  iqr = IQR(x, na.rm = FALSE)
  return(x - 1.5 * iqr, x + 1.5 * iqr)
}
```

```
## Error: <text>:2:16: unexpected symbol
## 1: whisker.endpoints = function(x) {
## 2:   if (is.na(x) warning
##                   ^
```

Syntax Errors

Run the above code chunk and read the errors. The first error that we get tells us there is an enexpected symbol - the 'w' in 'warning' is unexpected.

The problem is that R is expecting a closing parenthesis to surround the condition in the `if` statement. Copy the function from the previous code chunk into the chunk below and correct this problem.

```
whisker.endpoints = function(x) {
  if (is.na(x)) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = FALSE)
  iqr = IQR(x, na.rm = FALSE)
  return(x - 1.5 * iqr, x + 1.5 * iqr)
}
```

Now when you run this code chunk, you should not receive any errors.

Abnormal Termination

Next, test the funtion. Call it as follows.

```
whisker.endpoints(1:19)
```

```
## Warning in if (is.na(x)) warning("Careful x has NAs\n"): the condition has
## length > 1 and only the first element will be used
```

```
## Error in return(x - 1.5 * iqr, x + 1.5 * iqr): multi-argument returns are not permitted
```

When we test our code, we find that it abnormally terminates. That is, the code doesn't run.

The error message tells us there is a problem with our return from the function. It says 'multi-argument returns are not permitted.' We can only return one object from an R file.

Fix this problem, and retest your function.

```
whisker.endpoints = function(x) {
  if (is.na(x)) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = FALSE)
  iqr = IQR(x, na.rm = FALSE)
  return(c(x - 1.5 * iqr, x + 1.5 * iqr))
}
```

```
whisker.endpoints(1:19)
```

```
## Warning in if (is.na(x)) warning("Careful x has NAs\n"): the condition has
## length > 1 and only the first element will be used
```

```
##  [1] -12.5 -11.5 -10.5  -9.5  -8.5  -7.5  -6.5  -5.5  -4.5  -3.5  -2.5  -1.5
## [13]  -0.5   0.5   1.5   2.5   3.5   4.5   5.5  14.5  15.5  16.5  17.5  18.5
## [25]  19.5  20.5  21.5  22.5  23.5  24.5  25.5  26.5  27.5  28.5  29.5  30.5
## [37]  31.5  32.5
```

Our code now runs without abnormal termination.

Warning Message

When we call `whisker.endpoints(1:19)`, R gives us an error message. (We received this message earlier too.) Sometimes a warning message is not a problem, other times it points to a problem with our code.

The warning message that the condition in the if statement has a length greater than 1 and only the first element is used. Is this OK? Actually, we want to issue a warning message if any of the elements in `x` are `NA`. That is not what the code is doing. We can add the `any` function to the condition to fix this problem.

```
whisker.endpoints = function(x) {
  if (any(is.na(x))) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = FALSE)
  iqr = IQR(x, na.rm = FALSE)
  return(c(x - 1.5 * iqr, x + 1.5 * iqr))
}
```

Now, let's check our first test again.

```
whisker.endpoints(1:19)
```

```
##  [1] -12.5 -11.5 -10.5  -9.5  -8.5  -7.5  -6.5  -5.5  -4.5  -3.5  -2.5  -1.5
## [13]  -0.5   0.5   1.5   2.5   3.5   4.5   5.5  14.5  15.5  16.5  17.5  18.5
## [25]  19.5  20.5  21.5  22.5  23.5  24.5  25.5  26.5  27.5  28.5  29.5  30.5
## [37]  31.5  32.5
```

We no longer receive a warning message.

We need to test our function with more than one test case. Since our warning is for when `NA`s are present in the input, let's set up a test case for this situation.

```
whisker.endpoints(c(1:19, NA))
```

```
## Warning in whisker.endpoints(c(1:19, NA)): Careful x has NAs
```

```
## Error in quantile.default(x, probs = c(0.25, 0.75), na.rm = FALSE): missing values and NaN's not allo
```

We see that the warning appears, but now we have another error message. It tells us that `x`, the input variable has an `NA`. When we check our code, we see that we accidentally put `FALSE` instead of `TRUE` for the `na.rm` argument.

Help from `traceback()`

Suppose we didn't see the problem right away. We can get help in pinpointing where the error occurs by calling the `traceback()` function. We call traceback immediately after our call to `whisker.endpoints` and it prints the sequence of calls that led to the last error.

```
whisker.endpoints(c(1:19, NA))
```

```
## Warning in whisker.endpoints(c(1:19, NA)): Careful x has NAs
```

```
## Error in quantile.default(x, probs = c(0.25, 0.75), na.rm = FALSE): missing values and NaN's not allo
```

```
traceback()
```

```
## No traceback available
```

Unfortunately, when we knit the file, this information is not provided. If you run the above code chunks in your workspace, you will see the following printed to the console

```
traceback()
4: stop("missing values and NaN's not allowed if 'na.rm' is FALSE")
3: quantile.default(x, probs = c(0.25, 0.75), na.rm = FALSE)
2: quantile(x, probs = c(0.25, 0.75), na.rm = FALSE) at #3
1: whisker.endpoints(c(1:19, NA))
```

The most recent function call leading to the error is the call to `stop`, which stops the execution of the code. This function is called by `quantile.default`, which was called by `quantile`. Notice that the arguments to these function calls (`quantile` and `quantile.default`) are `x`, which we supply in our call as `c(1:19, NA)`, `probs`, and `na.rm`. We see that `na.rm = FALSE` which is the problem because it should be `TRUE`.

Fix this problem, and test again.

```
whisker.endpoints = function(x) {
  if (any(is.na(x))) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = TRUE)
  iqr = IQR(x, na.rm = TRUE)
  return(c(x - 1.5 * iqr, x + 1.5 * iqr))
}
```

Let's call our function with an input that includes `NA`.

```
whisker.endpoints(c(1:19, NA))
```

```
## Warning in whisker.endpoints(c(1:19, NA)): Careful x has NAs

##  [1] -12.5 -11.5 -10.5  -9.5  -8.5  -7.5  -6.5  -5.5  -4.5  -3.5  -2.5  -1.5
## [13]  -0.5   0.5   1.5   2.5   3.5   4.5   5.5    NA  14.5  15.5  16.5  17.5
## [25]  18.5  19.5  20.5  21.5  22.5  23.5  24.5  25.5  26.5  27.5  28.5  29.5
## [37]  30.5  31.5  32.5    NA
```

Now we get the warning message, but the code returns a value. It seems like our function works!

Silent Errors

By silent error, we mean an error in our code that goes undetected by R when we source our call our function. These errors come from mistakes that we make where the code is not computing what we want computed. The best way to uncover these errors is by developing an extensive set of tests for your functions.

We have a silent error in this code. If we examine the return value from the previous call we see that we are getting 42 numbers in return but want only 2.

Suppose, we are unsure of where the problem is coming from. One useful tool to help us debug, is the `browser` function.

A call to `browser()`

We can place a call to browser inside our function and when R reaches that expression, it puts us in the environment of the function where we can examine variables, change their values, and step through code. Place a call to `browser()` in your function immediately following the call to `quantile()`.

```
whisker.endpoints = function(x) {
  if (any(is.na(x))) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = TRUE)
  browser()
  iqr = IQR(x, na.rm = TRUE)
  return(c(x - 1.5 * iqr, x + 1.5 * iqr))
}

whisker.endpoints(c(1:19, NA))
```

Now call your function with the same input as before. Here's an example of how you might use the browser to check the computations within the function.

```
> whisker.endpoints(c(1:19, NA))
Called from: whisker.endpoints(c(1:19, NA)) ...
Browse[1]> ls()
[1] "qlu" "x"
Browse[1]> qlu
 25%  75%
 5.5 14.5
Browse[1]> head(x)
[1] 1  2  3  4  5  6
Browse[1]> n
debug at #6: iqr = IQR(x, na.rm = TRUE)
Browse[2]> n
debug at #7: return(c(x - 1.5 * iqr, x + 1.5 * iqr))
Browse[2]> ls()
[1] "iqr" "qlu" "x"
Browse[2]> iqr
[1] 9
Browse[2]> Q
```

Note the commands `n`, `c`, and `Q` respectively, evaluate the next expression, continue to the end of the current block (this may be the end of the function or the end of a sub-block in an if/else statement or loop), and quit the function.

We checked the code and see that `qlu` has 2 elements as expected and `iqr` is properly calculated. The problem must be with what we return. We see that the `return` uses `x` rather than `qul`. Fix this problem and test again.

```
whisker.endpoints = function(x) {
  if (any(is.na(x))) warning("Careful x has NAs\n")
  qlu = quantile(x, probs = c(0.25, 0.75), na.rm = TRUE)
  iqr = IQR(x, na.rm = TRUE)
  return(c(qlu[1] - 1.5 * iqr, qlu[2] + 1.5 * iqr))
}
```

Remember to remove the call to `browser`.

```
whisker.endpoints(1:19)
## 25% 75%
##  -8  28
whisker.endpoints(c(1:19, NA))
## Warning in whisker.endpoints(c(1:19, NA)): Careful x has NAs
## 25% 75%
##  -8  28
```

We can have the browser called automatically when an error occurs by setting up this option with

```
options(error = recover)
```

And, we can turn off this option with

```
options(error = NULL)
```

However, whenever you make a mistake a simple mistake, e.g., a typo, the `browser()` function will be called, which can get pretty annoying.