Debugging in R: tryCatch

Angela Zhang

11/7/2017

Debugging in R

Debugging is an often frustrating process of identifying and removing errors from a program. It's standard in any programming language and R is no exception. `tryCatch` is helpful for catching mistakes.

Figure 1: A humorous comic

Components of `tryCatch`

This function starts with `tryCatch()`. Within these parentheses, we have:

- A function that we want: i.e. `{function(x)}`
- If an error exists, we have a function that indicates we have an error. This is denoted by `error = {function(x)}`
- The `finally` argument executes all the expressions inside, regardless of whether an error occurred

Schematic: components of `tryCatch`

```r
testFunction <- function(x){
  tryCatch({(expr)}
           error = function(e){
             expr
             expr
           }
           finally = {
             expr # These expressions are run, regardless
                  # if there was an error
           })
}
```

Example of `tryCatch`

```
##   x y
## 1 1 1
## 2 2 2
## 3 3 7
## 4 5 9
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ x: num  1 2 3 5
##  $ y: Factor w/ 4 levels "1","2","7","9": 1 2 3 4
```

Here, `test` is a simple dataframe where the x column is numeric but the y column is a factor (as you can see by the `str` function)

Example of tryCatch

```r
meanTest <- function(x){
  suppressWarnings(
    tryCatch(colMeans(test),
             error= function(e){message("One column isn't numeric!")
                                test1 = apply(test, 2, as.numeric)
                                return(colMeans(test1))},
             finally={message("Changed all columns to numeric")}))
}


meanTest(test)
```

```
## One column isn't numeric!
```

```
## Changed all columns to numeric
```

```
##    x    y
## 2.75 4.75
```