

Debugging

Roger D. Peng, Associate Professor of Biostatistics

May 17, 2016

Something's Wrong!

Indications that something's not right - `message`: A generic notification/diagnostic message produced by the `message` function; execution of the function continues - `warning`: An indication that something is wrong but not necessarily fatal; execution of the function continues; generated by the `warning` function - `error`: An indication that a fatal problem has occurred; execution stops; produced by the `stop` function - `condition`: A generic concept for indicating that something unexpected can occur; programmers can create their own conditions

Something's Wrong!

Warning

```
log(-1)
```

```
## Warning in log(-1): NaNs produced
```

```
## [1] NaN
```

Something's Wrong

```
printmessage <- function(x) {  
  if(x > 0)  
    print("x is greater than zero")  
  else  
    print("x is less than or equal to zero")  
  invisible(x)  
}
```

Something's Wrong

```
printmessage <- function(x) {  
  if(x > 0)  
    print("x is greater than zero")  
  else  
    print("x is less than or equal to zero")  
  invisible(x)  
}  
printmessage(1)
```

```
## [1] "x is greater than zero"
```

```
printmessage(NA)
```

```
## Error in if (x > 0) print("x is greater than zero") else
```

Something's Wrong!

```
printmessage2 <- function(x) {  
  if(is.na(x))  
    print("x is a missing value!")  
  else if(x > 0)  
    print("x is greater than zero")  
  else  
    print("x is less than or equal to zero")  
  invisible(x)  
}
```

Something's Wrong!

```
printmessage2 <- function(x) {  
  if(is.na(x))  
    print("x is a missing value!")  
  else if(x > 0)  
    print("x is greater than zero")  
  else  
    print("x is less than or equal to zero")  
  invisible(x)  
}  
x <- log(-1)
```

```
## Warning in log(-1): NaNs produced
```

```
printmessage2(x)
```

```
## [1] "x is a missing value!"
```

Something's Wrong!

How do you know that something is wrong with your function? -
What was your input? How did you call the function? - What were you expecting? Output, messages, other results? - What did you get? - How does what you get differ from what you were expecting? - Were your expectations correct in the first place? - Can you reproduce the problem (exactly)?

Debugging Tools in R

The primary tools for debugging functions in R are - `traceback`: prints out the function call stack after an error occurs; does nothing if there's no error - `debug`: flags a function for “debug” mode which allows you to step through execution of a function one line at a time - `browser`: suspends the execution of a function wherever it is called and puts the function in debug mode - `trace`: allows you to insert debugging code into a function at specific places - `recover`: allows you to modify the error behavior so that you can browse the function call stack

These are interactive tools specifically designed to allow you to pick through a function. There's also the more blunt technique of inserting `print/cat` statements in the function.

traceback

```
> mean(x)
Error in mean(x) : object 'x' not found
> traceback()
1: mean(x)
>
```

traceback

```
> lm(y ~ x)
Error in eval(expr, envir, enclos) : object 'y' not found
> traceback()
7: eval(expr, envir, enclos)
6: eval(predvars, data, env)
5: model.frame.default(formula = y ~ x, drop.unused.levels
4: model.frame(formula = y ~ x, drop.unused.levels = TRUE)
3: eval(expr, envir, enclos)
2: eval(mf, parent.frame())
1: lm(y ~ x)
```

debug

```
> debug(lm)
> lm(y ~ x)
debugging in: lm(y ~ x)
debug: {
  ret.x <- x
  ret.y <- y
  cl <- match.call()
  ...
  if (!qr)
    z$qr <- NULL
  z
}
```

Browse[2]>

debug

```
Browse[2]> n
debug: ret.x <- x
Browse[2]> n
debug: ret.y <- y
Browse[2]> n
debug: cl <- match.call()
Browse[2]> n
debug: mf <- match.call(expand.dots = FALSE)
Browse[2]> n
debug: m <- match(c("formula", "data", "subset", "weights",
  "offset"), names(mf), 0L)
```

recover

```
> options(error = recover)
> read.csv("nosuchfile")
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'nosuchfile': No such file or directory

Enter a frame number, or 0 to exit

1: read.csv("nosuchfile")
2: read.table(file = file, header = header, sep = sep, quot
3: file(file, "rt")

Selection:
```

Debugging

Summary - There are three main indications of a problem/condition: message, warning, error - only an error is fatal - When analyzing a function with a problem, make sure you can reproduce the problem, clearly state your expectations and how the output differs from your expectation - Interactive debugging tools `traceback`, `debug`, `browser`, `trace`, and `recover` can be used to find problematic code in functions - Debugging tools are not a substitute for thinking!