Lecture 10: Debugging (1)

STAT598z: Intro. to computing for statistics

Vinayak Rao

Department of Statistics, Purdue University

Bugs

Bugs are an inevitable part of programming

Learning to fix them is an art that comes only from practice

(Un)fortunately, upto 90% programming time is debugging

- Try to avoid bugs

In the event of a bug:

- Locate the bug
- Understang what’s wrong
- Fix it

Different kinds of bugs

Syntax/run-time errors that cause fatal errors: Easiest to deal with

- Work backwards form the problematic line
- Find out what's expected and what's provided
- Read the manual
- google/stackexchange/your local R guru

Logical errors:

R runs without errors, however the output is wrong.

Much harder to track.

- Have you misunderstood some aspect of R functionality?
- Did you incorrectly translate your equations to code?
- Are your equations incorrect?
- Maybe your equations are correct but idea doesn’t work?

Programming style

Good programming style makes your life easier

Have informative variable/function names

Break your code into smaller functions and test individually.

Much better to build up from a set of bug-free components than to write down a big function and then debug.

- Much easier to deal with modular code
- Build your own library of functions/wrappers

Tracking down errors

Try to read the error message

Can be confusing, but is informative compared to e.g. Latex (gibberish) or C (usually ‘segmentation faults’)

```
"can’t find the object my_obj"
```

Have you set variable/loaded package?

Have you set variable/loaded package? A common error:

```
for(i in 1:10) a[i] <- i # First declare a!
```

Have you set variable/loaded package? A common error:

```
for(i in 1:10) a[i] <- i # First declare a!
```

```
"Error: Incompatible lengths ..."
```

What are the lengths?

```
missing value where TRUE/FALSE needed
```

What is the argument to if/ while ?

Tracking down errors

R always tell you where the error was detected

The actual cause can be much earlier than R indicates.

(In general, bad idea to write many lines of code without checking syntax a few times along)

A Google search often leads to a solution on stackexchange (be sure to remove variable names specific to your code)

‘Rubber-ducking’

(ericlippert.com/2014/03/05/how-to-debug-small-programs/)

- You should be able to explain in simple words why each line is correct (not necessarily to a rubber duck though)

For each line know what input and predicted outputs are

While debugging code, intersperse `if`'s and `print`'s.

```
if(prob > 1 || prob < 0) {
print ”NOOO!!! Invalid probability”
stop();
}
```

- compare expected with produced values

The `stopifnot` functions are also useful:

```
stopifnot(prob > 0, prob < 1)
```

Minimal working example

If you can’t track it down by inspection or want to email me:

- Create a Minimal Working Example (MWE)

A third person should:

- be able to reproduce error by cutting and pasting your code
- not have to worry about unnecessary details

Do not send me your entire code, saying ”Help”

Do not send me just the error message, saying ”Help”

Remove all unnecessary code after error (easy)

Remove all unnecessary code before error (harder):

- Remove unecessary variables/functions/packages
- Remove unnecessary layers in ggplot()
- If offending line is wrapped in a for/while loop, remove that
- If error involves a long vector, try to minimize its length
- Remove unnecessary columns in dataframes
- If random numbers are involved, set.seed()

Most class errors I’ve seen can be reduced to one assignment and one command

A good set of guidelines: http://stackoverflow.com/help/mcve (http://stackoverflow.com/help/mcve)

Write a minimal program:

- *Restart from scratch*: Starting from an empty files, add as few lines as possible to get your error
- *Divide-and-conquer*: Remove parts of program line by line till the error disappears, and add the last line back

Ask a specific question

"My code (see attached) gives an error": This is a story, not a question

“Why does my code (see attached) gives an error?”: Unhelpful question deserves unhelpful answer “Maybe your code is wrong”

Asking a specific question is halfway towards fixing your bug

Also convinces me that you thought about it