Debugging in R

Biostat 140.776
2004-10-04
Roger D. Peng

R tools for debugging

- `cat()`
- `print()`

Both can be used for printing messages to the screen, but `cat()` is more flexible.

Real R tools for debugging

- `traceback()`
- `debug()`
- `browser()`
- `trace()`
- `recover()`

`traceback()`

- When an R function fails, usually, an error is printed to the screen.
- Immediately after an error, you can call `traceback()` to see in which function the error occurred.

`debug()`

- Calling `debug()` on function `foo()` flags that function for “debugging”
  - e.g. `debug(foo)`
- When `foo()` is called in your program, execution will pause and you can step through `foo()` line by line.

`browser()`

- Inserting a call to `browser()` in a function will pause the execution of a function at the point where `browser()` is called.
- Similar to using `debug()` except you can control where execution gets paused.
trace()

• Calling `trace()` on a function allows the user to insert bits of code into a function
• The syntax for `trace()` is a bit strange for first time users
• You might be better off using `debug()`

recover()

• `recover()` can be used as an error handler, set using `options()`
  – e.g. `options(error = recover)`
• When a function throws an error, execution is halted at the point of failure
  – you can browse the function calls and examine the environment to find the source of the problem

General guidelines

• Don’t underestimate the usefulness of `print()` and `cat()`!
• `debug()` is your friend
• Try not to develop an unhealthy relationship with the debugger
• Think first, then program