Debugging is the process of getting your expectations to converge with reality. When writing software in any language, we develop a certain set of expectations about how the software should behave and what it should do. But inevitably, when we run the software, it does something *different* from what we expected. In these situations, we need to engage in a process to determine if

1. Our expectations were incorrect, based on the documented behavior of the software; or
2. There is a problem with the code, such that the programming is not done in a way that will match expectations.

This is the process of debugging.

In the previous section, we discussed what to do when software generates conditions (errors, warnings, messages) in a manner that is completely *expected*. In those cases, we know that certain functions will generate errors and we want to handle them in a manner that is not the usual way.

This section describes the tools for debugging your software in R. R comes with a set of built-in tools for interactive debugging that can be useful for tracking down the source of problems. These functions are

- browser(): an interactive debugging environment that allows you to step through code one expression at a time
- debug() / debugonce(): a function that initiates the browser within a function
- trace(): this function allows you to temporarily insert pieces of code into other functions to modify their behavior
- recover(): a function for navigating the function call stack after a function has thrown an error
- traceback(): prints out the function call stack after an error occurs; does nothing if there's no error