Stopping the execution of your program with stop() should only happen in the event of a catastrophe - meaning only if it is impossible for your program to continue. If there are conditions that you can anticipate that would cause your program to create an error then you should document those conditions so whoever uses your software is aware. Common failure conditions like providing invalid arguments to a function should be checked at the beginning of your program so that the user can quickly realize something has gone wrong. This is case of checking function inputs is a typical use of thestopifnot() function.

You can think of a function as kind of contract between you and the user: if the user provides specified arguments your program will provide predictable results. Of course it's impossible for you to anticipate all of the potential uses of your program, so the results of executing a function can only be predictable with regard to the type of the result. It's appropriate to create a warning when this contract between you and the user is violated. A perfect example of this situation is the result of as.numeric(c("5", "6", "seven")) which we saw before. The user expects a vector of numbers to be returned as the result of as.numeric() but "seven" is coerced into being NA, which is not completely intuitive.

R has largely been developed according to the Unix Philosophy (which is further discussed in Chapter 3) which generally discourages printing text to the console unless something unexpected has occurred. Languages than commonly run on Unix systems like C, C++, and Go are rarely used interactively, meaning that they usually underpin computer infrastructure (computers "talking" to other computers). Messages printed to the console are therefore not very useful since nobody will ever read them and it's not straightforward for other programs to capture and interpret them. In contrast R code is frequently executed by human beings in the R console which serves as an interactive environment between the computer and person at the keyboard. If you think your program should produce a message, make sure that the output of the message is primarily meant for a human to read. You should avoid signaling a condition or the result of your program to another program by creating a message.

Mark as completed