

Programming for Data Science

Debug in RStudio

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica



Debug in general

- Aim of debugging is to help you find bugs by figuring out where the code is not behaving in the way that you expect.

To do this, you need to:

- ▶ Begin running the code;
- ▶ Stop the code at the point where you suspect the problem is arising;
- ▶ Look at and/or walk through the code, step-by-step at that point.

Entering debug mode

- In order to enter debug mode, you need to tell R when you want to pause the computation.
 - ▶ Stopping on a line using **Editor breakpoints**;
 - ▶ Stopping when an error occurs using **Break in Code**.

Entering debug mode

Stopping on a line using **Editor breakpoints**:

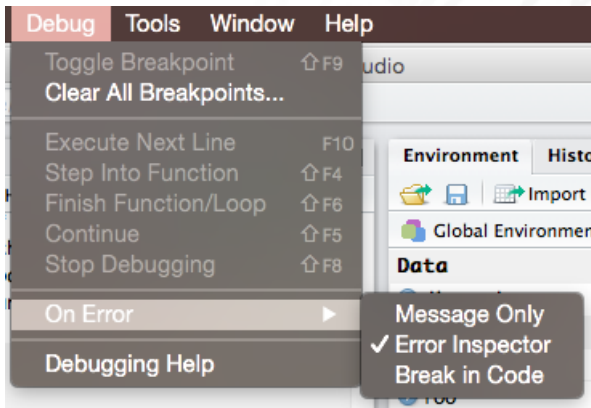
- The most common (and easiest) way to stop on a line of code is to set a breakpoint on that line. You can do this in RStudio by clicking to the left of the line number in the editor, or by pressing Shift+F9 with your cursor on the desired line.

```
18   best <- 0
19   for (x in 100:999) {
20     for (y in x:999) {
21       candidate <- x * y
22       if (candidate > best && palindrome(candidate)) {
23         best <- candidate
24       }
25     }
26   }
```

Entering debug mode

Stopping when an error occurs using **Break in Code** :

- If you are diagnosing a specific error, you can have RStudio halt execution at the point where the error is raised. To do this, go to Debug -> On Error and change the value from “Error Inspector” to “Break in Code”.



Debug in rstudio

Click next to line number to add/remove a breakpoint.

Highlighted line shows where execution has paused

Run commands in environment where execution has paused

Examine variables in executing environment

Select function in traceback in debug

The screenshot shows the RStudio IDE interface. The main editor window displays R code for a palindrome function and a function to find the largest palindrome product. The console shows the execution of the function, which has paused at line 12. The environment pane on the right shows the current environment with variables: digit1 (0), digits (5), num (10000L), and x (1L). The traceback pane shows the call stack, with the current function call highlighted. The console shows the following commands and output:

```
9:1 palindrome(num)
Browse[3]> f
debug at ~/RStudio-Essentials/Essentials-2/palindrome.R#12: digit2
<- get_digit(num, (digits + 1) - x)
Browse[2]>
```

The environment pane shows the following values:

Variable	Value
digit1	0
digits	5
num	10000L
x	1L

The traceback pane shows the following stack:

```
palindrome(candidate) at palindrome.R:12
biggest_palindrome() at palindrome.R:25
```

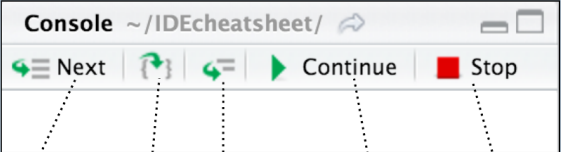
Debug in rstudio

Launch debugger mode from origin of error

Open traceback to examine the functions that R called before the error occurred



```
Console ~/IDEcheatsheet/ ↗  
> foo()  
Error in get_digit(num, x) :  
Error!  
Show Traceback  
Rerun with Debug
```



Next | Step into and out of functions | Continue | Stop

Step through code one line at a time

Step into and out of functions to run

Resume execution

Quit debug mode

Exercise: Find the BUG

```
>BUG=function(spin){  
  N = length(spin)  
  spin = c(spin, sum(spin), mean(spin))  
  if (sum(is.na(spin)) != 0){  
    cat("Error")  
    return(NA)  
  }  
  for (i in 1 : N + 2){  
    spin[i] = spin[i] + spin[i]  
  }  
  return(spin)  
}
```

Expected behavior: BUG(1:4) → 2 4 6 8 20 5

Expected behavior: BUG(c(1,3,5,6,NA)) → NA